University of Zurich[UZH]

Using Biometric Sensors to Increase Developers' Productivity

Dissertation submitted to the Faculty of Business, Economics and Informatics of the University of Zurich

to obtain the degree of
Doktor der Wissenschaften, Dr. sc.
(corresponds to Doctor of Science, PhD)

presented by
Sebastian C. Müller
from
Vaduz, Liechtenstein

approved in April 2016

at the request of
Prof. Dr. Thomas Fritz, University of Zurich, Switzerland
Prof. Dr. Emerson Murphy-Hill, NC State University, USA
Prof. Dr. Harald C. Gall, University of Zurich, Switzerland

2016

The Faculty of Business, Economics and Informatics of the University of Zurich hereby authorizes the printing of this dissertation, without indicating an opinion of the views expressed in the work.

Zurich, April 6, 2016

Chairwoman of the Doctoral Board:
Prof. Dr. Elaine M. Huang

Acknowledgements

It is my pleasure to thank all the people who made this thesis possible.

First and foremost, I would like to thank my advisor, Thomas Fritz, for his continuous guidance and support during my doctoral studies. No matter how busy he was or where in the world he was working, he always found time when I was looking for advice.

I'm also especially grateful to Harald Gall. He made this thesis possible in the first place, by giving me the opportunity to work in his group. He also agreed to be part of my PhD committee and spent his precious time to evaluate my thesis.

Special thanks go to Emerson Murphy-Hill for being part of my PhD committee as an external examiner and for dedicating his valuable time to evaluate my work.

I appreciate the feedback that Andrew Begel gave me about my research throughout my studies, in particular the frequent discussions we had about how to collect and analyze biometric data.

During a PhD there are many ups and downs. It would not have been possible to make it through all the downs and celebrate the ups without my fellow members of the SEAL group. It was an awesome experience to work with you all. A lot of thanks to Amancio Bouza, André Meyer, Carol Alexandru, Chaman Wijesiriwardana, Christian Inzinger, Emanuel Giger, Gerald Schermann, Giacomo Ghezzi, Jürgen Cito, Katja Kevic, Manuela Züger, Martin Brandtner,

Matthias Hert, Michael Würsch, Philipp Leitner, and Sebastiano Panichella for the fun time we had in the office and for their valuable feedback on my research.

Last but not least, I want to thank my girlfriend, Martina, and my parents, Judith and Roland, for their unconditional support during my studies.

Sebastian Müller
Zürich, April 2016

Abstract

The development of software is a cost- and people-intensive process. For years, the software development industry has been coping with a shortage of software developers. Besides just training even more software developers, an alternative and particularly promising way to tackle this problem, is to boost the productivity of every single developer. Traditionally, research on developers' productivity has primarily focused on assessing their output using certain metrics and has therefore suffered from two major drawbacks: most of these approaches do not take into account the individual differences that exist between software developers, and the metrics used for these approaches can, in most cases, only be calculated once the work is done.

Emerging biometric sensors offer a new opportunity to gain a better understanding of what developers perceive during their work and thereby a new way to better understand what aspects are affecting developers' productivity. The basic idea behind biometric sensing is to measure a person's physiological features that in turn can be linked to a person's psychological states. A multitude of studies in psychology have already shown that biometric measurements can be used to assess the emotional and cognitive states of a developer.

In our research, we investigate the use of biometric measurements to assess a developer's perceived difficulty, progress and emotions while working on a change task. Based on the assumption that more difficult code has a higher likelihood to contain a bug compared to code that is perceived as being easier, we also investigate the use of biometric measurements to identify code quality concerns in the code developers are changing. Our vision is to gain a better understanding

of what every individual developer experiences, feels or perceives during his/her work, and how these aspects affect his/her productivity, to suggest approaches which increase every individual developer's productivity.

In our research, we conducted three studies, ranging from lab experiments to a two-week field study, to investigate the use of biometric sensors in a software development context. The results of our studies provide initial evidence that biometrics can be used to better understand what a developer perceives in real-time, while s/he is working on a change task. In particular, using biometric data, we were able to distinguish between positive and negative emotions, phases of high and low progress and to predict a developer's perceived difficulty while working on a change task with high accuracy. Additionally, we were able to use biometrics to predict code quality concerns that were identified in peer code reviews. These findings open up many opportunities for better supporting developers in their work, for instance by automatically and instantaneously detecting potential quality concerns in the code, before they are committed to the code repository, or by avoiding costly interruptions when a developer is in the flow and making a lot of progress.

Zusammenfassung

Die Entwicklung von Software ist ein kosten- und arbeitsintensiver Prozess. Trotz der wachsenden Anzahl an Softwareentwicklern kämpft die Software-Industrie seit Jahren mit einem Mangel an Softwareentwicklern. Neben der Ausbildung von weiteren Softwareentwicklern liegt eine besonders vielsprechende Lösungsmöglichkeit darin, die Produktivität jedes einzelnen Softwareentwicklers zu steigern.

Forschung mit dem Ziel die Produktivität eines Entwicklers zu steigern fokussierte sich bis jetzt vor allem auf die Artefakte, beispielsweise Source Code, die Entwickler produzieren, und hat versucht diese Artefakte mit Hilfe von verschiedenen Metriken zu messen. Diese Vorgehensweise hat zwei Nachteile: erstens können die individuellen Unterschiede, die zwischen Entwicklern existieren, nicht in Betracht gezogen werden, und zweitens können die Metriken, die für diese Vorgehensweise verwendet werden, oft erst berechnet werden, nachdem die Arbeit an einer Source Code Änderungsaufgabe (Change Task) vollständig abgeschlossen ist.

Neuerdings bieten biometrische (auch bekannt als psychophysiologische) Sensoren die Möglichkeit besser zu verstehen was ein Softwareentwickler während seiner/ihrer Arbeit empfindet und welche Aspekte seine/ihre Produktivität am Arbeitsplatz beeinflussen. Verschiedene Studien im Bereich der Psychologie haben bereits gezeigt, dass biometrische Sensoren verwendet werden können, um verschiedene Emotionen einer Person, als auch einige ihrer kognitiven Empfindungen, beispielsweise Cognitive Load (CL), zu bestimmen.

Für unsere Forschung untersuchen wir die Realisierbarkeit biometrische Sensoren zu verwenden, um die empfundenen Schwierigkeiten, den Fortschritt und

die Emotionen eines Entwicklers, während dieser an einer Source Code Änderungsaufgabe arbeitet, zu ermitteln. Dieselben biometrischen Messwerte wollen wir verwenden um Stellen im Code zu finden, die eine mangelnde Software Qualität aufweisen. Grundlage dafür ist die Vermutung, dass Code, welcher von Entwicklern als schwierig empfunden wird, eine höhere Wahrscheinlichkeit hat einen Fehler zu enthalten, als einfacher Code. Unser langfristiges Ziel ist es, die Aspekte, welche einen Entwickler bei seiner/ihrer Arbeit erfährt, empfindet oder wahrnimmt, zu identifizieren, und besser zu verstehen, wie diese seine/ihre Produktivität beeinflussen. Dieses verbesserte Verständnis wollen wir anwenden, um jeden einzelnen Entwickler effizienter in seiner/ihrer Arbeit zu unterstützen und dadurch langfristig die individuelle Produktivität zu steigern.

Wir haben drei verschiedene Studien, die von einfachen Experimenten bis zu einer mehrwöchigen Feldstudie reichen, durchgeführt, um die Realisierbarkeit von biometrischen Sensoren im Bereich der Software Entwicklung zu untersuchen. Die Resultate dieser Studien zeigen, dass wir biometrische Sensoren verwenden können, um in Echtzeit die Empfindungen eines Entwicklers, der an einer Source Code Änderungsaufgabe arbeitet, besser verstehen zu können. Insbesondere ist es möglich die biometrischen Sensoren zu verwenden um zwischen positiven und negativen Emotionen, sowie zwischen Phasen von grossem und kleinem Fortschritt an einer Änderungsaufgabe, mit grosser Genauigkeit bestimmen zu können. Ausserdem konnten wir die Sensoren verwenden um die empfundene Schwierigkeit eines Entwicklers vorherzusagen und potentielle Stellen im Code mit mangelnder Qualität zu bestimmen. Diese neuen Erkenntnisse bieten nun viele Möglichkeiten, um einen Entwickler bei seiner/ihrer Arbeit besser unterstützen zu können. Zum Beispiel könnten mögliche Qualitätsprobleme im Code automatisch und frühzeitig erkannt werden, noch bevor ein Entwickler den mangelhaften Code in einem Code Repository speichert, oder Unterbrechungen durch Mitarbeiter könnten nach Möglichkeit verhindert werden, falls sich ein Entwickler gerade im Flow befindet und viel Fortschritt macht.

Contents

List of Figures

List of Tables

Acronyms

AUC area under the curve

BCI brain-computer interface

BVP blood volume pulse

CBF cerebral blood flow

CL cognitive load

CLT cognitive load theory

EDA electro-dermal activity

EEG electroencephalogram or electroencephalography

EMG electromyogram

EOG electrooculography

ER-SCR event-related skin conductance response

fMRI functional magnetic resonance imaging

GB gigabyte

GSR galvanic skin response

HR heart rate

HRV heart rate variability

IDE integrated development environment

NIRS near infrared spectroscopy

NS-SCR non-specific skin conductance response

P precision

PNS parasympathetic nervous system

PPG photoplethysmography

PSP personal software process

QC quality concern

R recall

RMSSD root mean square of successive differences

RQ research question

RR respiratory rate

SC skin conductance

SCL skin conductance level

SD standard deviation *or* software development

SDK software development kit

SNS sympathetic nervous system

TLX task load index

1

Synopsis

The development of software is a cost- and people-intensive process [Ephrath, 1988], [Boehm et al., 2005]. For years, the software development industry has been coping with a shortage of software developers [European Commision, 2013], [University of Phoenix, 2015]. To tackle this problem, one can either train even more software developers, or, more promisingly, make each existing software developer more productive. This raises some intriguing unanswered questions: What does it really mean for an individual developer to be productive? How are developers doing their work, what is going on in their minds and when are they experiencing difficulties?

Extensive research in the area of software engineering has examined how to help developers to be more productive. Traditionally, researchers aiming to increase developers' productivity have focused on what developers have done, measuring their output and collecting data from software repositories. For instance, several approaches have been proposed to automatically determine the defect-proneness of code based on metrics, such as code complexity or code

churn [Nagappan and Ball, 2005], [Zhang et al., 2007]. While these approaches can help reduce cost and effort in the software development process, they generally suffer from two major drawbacks: the metrics used for these approaches can in most cases only be calculated after change tasks are completed, and they do not take into account the individual differences that exist between developers, such as the differences between novice and experienced developers [Crosby and Stelovsky, 1990]. One of the few approaches that focused more on the individual developer is Humphrey's Personal Software Process (PSP) [Humphrey, 1996]. PSP provides developers with ways to measure, understand and improve their performance. However, the required manual tracking of various measures is tedious, time-consuming and only allows for coarse-grained insights.

To really help developers be more productive, we first need to gain a better understanding of what developers experience during their work and how these aspects affect their productivity. Emerging biometric (*aka.* psychophysiological) sensors in combination with research in psychology offer new ways to focus on and measure more of an individual developer, such as his/her cognitive and emotional states, while s/he is working on a task. The basic idea behind biometric sensing is to measure a person's physiological features that in turn can be linked to a person's psychological states. As an example, a person who is stressed generally tends to sweat more than in less stressful situations and this difference in sweat leads to a varying electronic conductance of the skin that can be measured by electro-dermal activity (EDA) sensors. In psychology, biometric measurements have already been used extensively and studies have shown that they can be used to quantify a person's emotional states (positive and negative, as well as specific emotions) or cognitive load (the amount of mental resources needed to perform a task).

In our research, we aim to investigate the use of biometric sensors in the context of software engineering to gain a better understanding of the cognitive and emotional states that individual developers experience during work. In particular, we investigate the use of biometric measurements to determine developers' perceived difficulty, progress, and emotions while working on a change task, as well as to identify potential quality concerns in code elements. In the long term,

we aim to use this improved understanding to better support developers in their work and increase every individual developer's productivity. In the following, we briefly describe each software development aspect that we investigate in our research.

Developers' Perceived Difficulty. While working on change tasks, developers experience various difficulties. For instance, they might have difficulties to understand the relevant code for a task, comprehend the design of the underlying software system, or acquire all the information needed for the task at hand [Ko et al., 2004], [Sillito et al., 2005]. Knowing when a developer experiences difficulties while working on a change task could help to better support developers in their work. For instance, we could provide additional support to developers who experience a lot of difficulties or we might be able to prevent particularly expensive interruptions in these phases. Research in psychology has already shown that certain biometric features are linked to a person's mental effort for working on a task. In our research, we examine the use of biometrics to assess a developer's perceived difficulty while working on a change task.

Code Quality Concerns. Based on developers' perceived difficulties while working on a change task, we might also be able to identify quality concerns (*e.g.*, bugs) while a developer is making changes to the code. Research has shown that with an increasing cognitive load — the amount of mental resources that are required to perform a task [Sweller, 1988] — people typically make more mistakes [Weast and Neiman, 2010], [Lavie et al., 2004], [Ayres, 2001], [Leppink and van den Heuvel, 2015]. The amount of cognitive load that is perceived depends on personality traits, but also on task characteristics, such as task difficulty and format, and can be assessed by biometrics [Haapalainen et al., 2010], [Wilson, 2002], [Veltman and Gaillard, 1998]. Based on the correlation between difficulty, cognitive load and error rate, we assume that a developer who experiences a lot of difficulties has a higher chance to create a bug than a developer who experiences less difficulties. In our research, we then examine whether we are able to use biometrics to identify code quality concerns — using cognitive load as a proxy — and prevent bugs from even being committed to the code repository.

Developers' Emotions & Progress. During their work, developers typically perceive a wide range of emotions, ranging from happiness or joy all the way to frustration or anger. Often these emotions are intertwined with the progress they make, for instance, making no progress on a task leading to frustration [Burleson and Picard, 2004]. Being able to determine a developer's emotions and progress while working on a task could open up many opportunities for improving a developer's productivity, for instance, by providing additional support or personalized recommendations when a developer is currently stuck and cannot make any progress on a particular task. Research has shown that biometrics might be able to detect and distinguish between emotions [Picard et al., 2001] [Reuderink et al., 2009], as well as to determine a flow or stuck state [Burleson and Picard, 2004], [Muldner et al., 2010], at least when working on small analytical task. In our research, we investigate the applicability of biometrics to predict perceived emotions and progress during realistic software development tasks.

In summary, we investigate the use of biometric measurements to gain a better understanding of developers' cognitive and emotional states during change tasks. The underlying hypothesis of our research is:

A combination of biometric sensors can be used to determine a software developer's perceived difficulty, progress and emotions when working on change tasks, as well as to predict potential code quality concerns in code elements.

To investigate this hypothesis, we conducted several studies. These studies ranged from controlled lab experiments to a two-week long field study. In particular, we focused on three different research questions that are described in more detail in the next chapter (Chapter 1.1). Chapter 1.2 provides an overview of the psychological and psychophysiological foundations of our research. The technical approach we used to answer our research questions is described in Chapter 1.3. The findings of our research are presented in Chapter 1.4, followed by the threats to validity (Chapter 1.5). Then we discuss challenges

(Chapter 1.6) and potential future work (Chapter 1.7) of our work, discuss related work (Chapter 1.8), summarize our work and contributions (Chapter 1.9) and describe the remainder of this thesis (Chapter 1.10).

1.1 Research Questions

Our research on using biometrics to increase a developer's productivity is divided into the following three research questions:

(RQ1) Can biometric measurements, such as electro-dermal activity or an electroencephalogram, be used to predict developers' perceived difficulty while working on change tasks?

(RQ2) Can biometric measurements be used to identify quality concerns in code elements in a real world context?

(RQ3) Can biometric measurements be used to determine a developer's emotions and perceived progress during a change task?

Table 1.1 provides an overview of these three research questions and the context in which we answered each research question. The first research question was investigated by two studies: a lab study (Chapter 2) and a field study (Chapter 3). For the lab study, we were able to recruit 15 professional software developers who worked on small code comprehension tasks. For the field study, we had 15 professional software developers working on their usual changes tasks in their normal work environment. With this field study, we also addressed our second research question, with a subset of 10 professional software developers. Finally, to find answers to the third research question, we conducted a lab study (Chapter 4) with 6 professional software developers and 11 PhD students with a major in computer science. The study participants had to work on study tasks representative of change tasks professional software developers work on.

The studies to address our research questions build upon each other and add incrementally more evidence of the use of biometrics in a software development context. First, we started with a lab study with small code comprehension

Table 1.1: Overview of the context in which we answered each research question (*Prof.* is referring to professional software developers and *Stud.* to PhD students with a major in computer science.).

RQ	Topic	Lab	Field	#Prof.	#Stud.	Study tasks	Chapters
1	Perceived difficulty	✓	✓	30		small & real tasks	2 & 3
2	Quality concerns		✓	10		real tasks	3
3	Emotions & progress	✓		6	11	tasks representative of real ones	4

tasks, before we moved to a lab study with longer tasks representative of real change tasks. After that, we conducted a field study investigating developers working on their usual tasks and in their usual work environment. Starting with lab experiments allowed us to limit the noise that can affect biometric sensors substantially and to get an initial idea of the value of biometrics. Switching to the field allowed us to then also investigate the value of biometrics in a realistic setting.

1.2 Psychological and Psychophysiological Foundations

Since the early 20$^{\text{th}}$ century, researchers have been investigating how processes in the body react to changing mental states [Cacioppo et al., 2007]. For instance, researchers have examined how the human body reacts when people are solving problems, experiencing stress, or processing information, and called the study of these correlations psychophysiology [Andreassi, 2007]:

> *Psychophysiology is the study of relations between psychological manipulations and resulting physiological responses, measured in the living organism, to promote understanding of the relation between mental and bodily processes.*

To quantify physiological (bodily) processes, researchers have used a variety of methods, measurements and devices, such as an electroencephalogram (EEG)

to measure brain activity, electromyograms (EMG) to measure muscle activity or pupillometry to measure changes in the pupil size. Using these methods, researchers have found correlations between, for instance, cognitive load and eye measurements [Haapalainen et al., 2010], [Klingner, 2010], [Ikehara and Crosby, 2005], or emotions and electro-dermal activity [Ax, 1953], [Greco et al., 2012], [McDuff et al., 2012], [Setz et al., 2010].

In our research, we aim to take advantage of these insights. We build our work on top of established correlations between psychophysiological measurements and various outcome measures, and apply them in the context of software engineering. In particular, we investigate the use of biometrics to measure a developer's perceived difficulty, progress and emotions. Figure 1.1 provides an overview of the relationships between the biometric measurements we use in our research, the underlying psychological concepts and the outcome measures we investigate in our research. As illustrated in Figure 1.1, our research is based on two distinct psychological concepts: cognitive load and the concept of arousal and valence. These two concepts are described in detail in the following two sections, while the biometric features to capture these concepts are presented in Chapter 1.2.3.

1.2.1 Cognitive Load

Cognitive load (CL) refers to the total amount of mental effort that is needed to perform a task. There are three different types of cognitive load whose sum builds the total amount of cognitive load imposed on an individual working on a task: intrinsic cognitive load, extraneous cognitive load, and germane load. The intrinsic cognitive load is caused by the intrinsic characteristics of the information that needs to be processed while working on the task, while the extraneous cognitive load is posed by the form in which this information is presented. Finally, the germane load refers to the amount of effort that is put into creating a permanent knowledge of the task that has been worked on [Sweller et al., 2011]. It is important to note that cognitive load differs between individuals, *i.e.*, working on the same task does not yield the same cognitive load level on every individual [Plass et al., 2010], since the experienced cognitive load level can be influenced by a person's age, experience and personality traits.

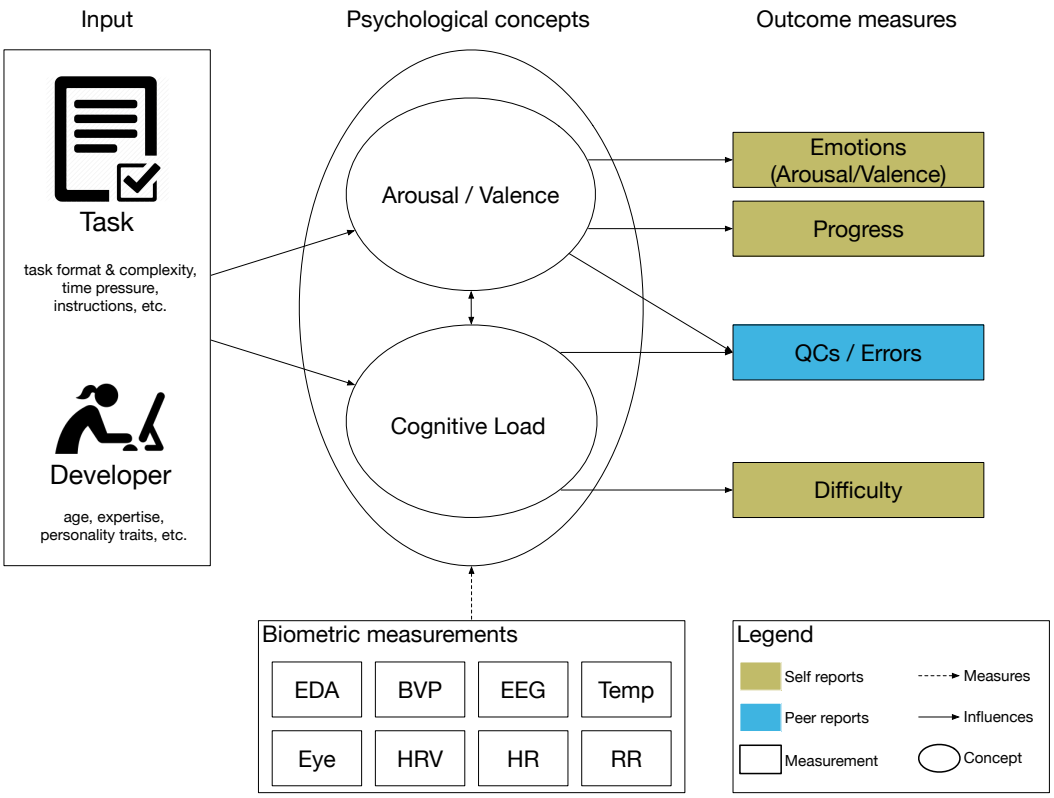


Figure 1.1: Relationship between biometric measurements, cognitive load, arousal / valence and the outcome measures of our research.

Various studies have shown that high cognitive load in general decreases an individual's performance and typically leads to a higher error rate [Weast and Neiman, 2010], [Lavie et al., 2004], [Ayres, 2001], [Leppink and van den Heuvel, 2015].

Other studies in psychology have shown that biometric measurements can be used to determine cognitive load [Haapalainen et al., 2010], [Veltman and Gaillard, 1998]. Based on the link to cognitive load, we might be able to use biometric measurements to determine a developer's perceived difficulty and the probability of an error being created.

1.2.2 Arousal & Valence

Research on emotions has a long history and many different theories and terminologies have been introduced. A widely accepted approach by Russell [Russell, 1980] differentiates between two cognitive dimensions of emotions: pleasure-displeasure and arousal-sleep. Today, these two dimensions are commonly called valence and arousal [Russell, 2003]. While the valence dimension is considered as the positive or negative character of an emotion [Colombetti, 2005], the arousal dimension indicates the amount of activation and excitement associated with an emotion [Russell, 2003]. A person's emotions can be influenced by various factors, such as task difficulty or personality traits [Basch and Fisher, 1998], [Plutchik and Conte, 1997], [Hutt and Weidner, 1993]. Similar to the correlation between cognitive load and specific outcome measures, there are correlations between arousal/valence and an individual's perceived progress and emotions [Brief and Weiss, 2002], [Graziotin et al., 2013], [Khan et al., 2011].

Various studies in psychology have shown that biometric measurements can be used to determine valence and arousal [Murugappan et al., 2008], [Murugappan et al., 2009], [Chanel et al., 2008]. We therefore might be able to use biometric measurements to determine emotions and progress, using the arousal/valence concept as a proxy.

1.2.3 Biometric Measurements

Based on the physiological origin, biometric measurements that we used in our work can roughly be categorized into eye-related, brain-related, skin-related, heart-related and breathing-related measurements. Each category is briefly explained in the following sections. Table 1.2 provides an overview of the measurements we used in our research together with the aspects that these measurements have been linked to in literature.

Eye-related Measurements. In general, we used three different types of eye-related measurements in our studies: eye movements, pupil size and eye blinks. All these measurements can be captured with an eye tracking device. The process of measuring eye movements is called electrooculography (EOG). There are two

Table 1.2: Overview of biometric measurements and the aspects related to them in literature (effects in blue are related to valence/arousal and effects in green to cognitive load).

Measurement	Previously found links
Eye-related	
Pupil size	Cognitive/memory/mental load [Haapalainen et al., 2010], [Klingner, 2010], [Beatty, 1982], [Iqbal et al., 2004]; Excitement [Muldner et al., 2010]; Positive and negative affect [Muldner et al., 2009]
Saccades	Mental workload [Brookings et al., 1996]; Evaluation of user interfaces [Goldberg and Kotval, 1999]
Fixations	Cognitive load [Ikehara and Crosby, 2005]; Performance during code reviews [Uwano et al., 2006, Sharif et al., 2012]; Effort to identify variable identifiers [Sharif et al., 2012]; Valence [Carniglia et al., 2012]
Brain-related	
Eye blinks	Visual attention/demand [De Jong and Merckelbach, 1990], [Wilson, 2002]; Mental workload [Brookings et al., 1996], [Ryu and Myung, 2005]; Frustration [Kapoor et al., 2007]; Stress and anxiety [Doehring, 1957]
Frequency bands (Alpha, Beta, Gamma, Delta, Theta)	Mental workload [Brookings et al., 1996], [Ryu and Myung, 2005]; Cognitive task classification [Lee and Tan, 2006]; Auditory awareness [Makeig and Jung, 1996]; Valence [Reuderink et al., 2013], [Sammler et al., 2007]; Arousal [Reuderink et al., 2013]; Happiness and sadness [Li and Lu, 2009]; Various emotions [Murugappan et al., 2008], [Murugappan et al., 2009]
Ratios of frequency bands	Memory load [Grimes et al., 2008]; Task engagement index [Kramer, 1991], [Lawrence J. Prinzel et al., 2001], [Berka et al., 2007]; Car driver status in various conditions [Brookhuis and De Waard, 1993]; Valence and arousal [Lin et al., 2010]
Attention and Meditation	Cognitive load [Haapalainen et al., 2010]; Valence and arousal [Yoon et al., 2013]
Skin-related	
Electro-dermal activity	Anger and fear [Ax, 1953]; Arousal and engagement [McDuff et al., 2012]; Distinguish stress from cognitive load [Setz et al., 2010]; Valence and arousal [Haag et al., 2004], [McDuff et al., 2012], [Leite et al., 2013], [Drachen et al., 2010]; Frustration [Scheirer et al., 2002], [Kapoor et al., 2007], [Mandryk et al., 2006], [Freeman, 1940]; Various emotions [Picard et al., 2001], [Ekman et al., 1983], [Chanel et al., 2008], [Wagner et al., 2005], [Greco et al., 2012]; Mental load [Wilson, 2002], [Richter et al., 1998]; Task difficulty [Novak et al., 2010], [Cornforth et al., 2015]
Skin temperature	Valence and arousal [Haag et al., 2004]; Boredom, engagement and anxiety [Chanel et al., 2008]; Various emotions [Ekman et al., 1983]; Task difficulty [Anthony et al., 2011], [Novak et al., 2010], [Cornforth et al., 2015]
Heart-related	
Blood volume pulse	Frustration [Scheirer et al., 2002]; Various emotions [Picard et al., 2001]; Valence and arousal [Haag et al., 2004]
Heart rate variability	Anxiety [Rani et al., 2004]; Various emotional states [McCraty and Tomasino, 2006]; Mental effort/load [Veltman and Gaillard, 1998], [Wilson, 2002], [Richter et al., 1998]; Task difficulty [Walter and Porges, 1976]; Task demand [Fairclough et al., 2005]
Heart rate	Valence [Sammler et al., 2007], [Haag et al., 2004]; Arousal [Haag et al., 2004]; Positive and negative affect [Drachen et al., 2010]; Happiness [Steptoe et al., 2005]; Various emotions [Wagner et al., 2005], [McCraty and Tomasino, 2006], [Chanel et al., 2008]; Mental effort/load [Veltman and Gaillard, 1998] [Wilson, 2002], [Richter et al., 1998]; Task difficulty [Walter and Porges, 1976], [Anthony et al., 2011], [Cornforth et al., 2015]
Breathing-related	
Respiratory rate	Mental effort [Veltman and Gaillard, 1998]; Task difficulty [Kuznetsov et al., 2011]; Task demand [Fairclough et al., 2005]

important kinds of eye movements for our research. The *fixation* on a stationary object, and the *saccade* from one object to another [Woodworth, 1948]. Saccades typically last for only a fraction of a second [Andreassi, 2007], depending on how far away the two fixation points are. On the other hand, fixations can last for seconds, depending on how long the eye fixates on the stationary object. In software engineering research, these measurements are predominantly used to assess the usability of user interfaces (*e.g.*, [Poole and Ball, 2006]), but in psychology, they were also linked to mental and cognitive load [Brookings et al., 1996], [Ikehara and Crosby, 2005].

The pupil size depends on many different factors such as the age, the gender or the illuminance. However, in general, the pupil diameter varies between 2mm and 4mm in bright light and 4mm and 8mm in darkness [Spector, 1990]. The pupil diameter may also vary when an individual perceives high mental load [Haapalainen et al., 2010] or is excited [Muldner et al., 2010].

Finally, the average eye blink rate is between 15 and 20 blinks per minute in a relaxed state [Andreassi, 2007]. When reading, the blink rate can be as low as three blinks per minute, while it can increase significantly when an individual is tired, experiences stress or is under time pressure [Andreassi, 2007].

All eye-related measurements have predominantly been used in research to measure cognitive and memory load [Brookings et al., 1996], [Ikehara and Crosby, 2005], [Goldberg and Kotval, 1999], [Sharif et al., 2012], [Uwano et al., 2006], [Haapalainen et al., 2010], [Iqbal et al., 2004], [Klingner, 2010]. In our research, we used the number and durations of fixations and saccades, as well as the pupil size and the eye blink rate as features in our machine learning experiments.

Brain-related Measurements. The electrical activity of the brain is caused by specific brain cells, called neurons [Andreassi, 2007]. The varying degree of activity of these neurons causes fluctuations in the voltage potential along the scalp that can be captured by an electroencephalography. From data captured with an EEG, research has identified various frequency bands (*aka.* brain wave patterns) that are called alpha (α), beta (β), gamma (γ), delta (δ), and theta (θ). Each of these frequency bands has a specific amplitude and each one has been

linked to various cognitive and emotional states. Alpha waves typically occur when an individual is in a relaxed state. As soon as the mental or physical activity level increases, the alpha waves disappear or their amplitude gets significantly smaller. Beta waves occur when performing mental or physical activities and gamma waves typically appear as a reaction to a sensory stimuli. In healthy humans, delta waves are only recognizable during deep sleep. Finally, theta waves appear when an individual experiences pleasure or displeasure [Andreassi, 2007]. In general, brain wave patterns have been correlated in research with mental workload and task difficulty [Brookings et al., 1996], [Ryu and Myung, 2005], [Kramer, 1991], [Lee and Tan, 2006]. For most of our studies, we used these frequency bands and ratios thereof as features.

Skin-related Measurements. In our research, we focused on two skin-related measurements: the electro-dermal activity and the skin temperature. The electro-dermal activity, formerly known as galvanic skin response (GSR) or skin conductance (SC), measures the electrical conductance of the skin. The variance in the electrical conductance of the skin is mainly caused by sweating. Sweating is controlled by the sympathetic nervous system (SNS) and when the SNS is highly aroused, the sweat glands in the skin will produce more sweat and in turn the electrical skin conductance will increase [Boucsein, 2012]. The EDA signal consists of two parts: the slowly changing, low frequency *tonic* part, and the fast adapting, high frequency *phasic* part [Schmidt and Walach, 2000]. Research has shown that electro-dermal activity correlates particularly strongly with arousal [Boucsein, 2012], [Dawson et al., 2007], [Schmidt and Walach, 2000], and emotions in general [Haag et al., 2004], [McDuff et al., 2012], [Scheirer et al., 2002], [Kapoor et al., 2007], [Mandryk et al., 2006], [Freeman, 1940], [Leite et al., 2013], [Picard et al., 2001], [Ekman et al., 1983], [Chanel et al., 2008], [Wagner et al., 2005].

In a healthy human, the core body temperature lies between 36.5 °C and 37.5 °C [Dinarello and Porat, 2012] and the mean skin temperature around 33 °C [Blatteis, 1998]. Research has shown that the skin temperature is closely linked to emotions [Collet et al., 1997], [Ekman et al., 1983], [Levenson et al., 1990]. In our studies, we used features describing the peaks in the temperature

signal, the mean value of the temperature signal, features referring to the peaks in the phasic part of the EDA signal, and commonly used features such as the mean value or the area under the curve (AUC) from the tonic part of the EDA signal.

Heart-related Measurements. In our studies, we used three different heart-related measurements: the blood volume pulse (BVP), the heart rate (HR) and the heart rate variability (HRV). The blood volume pulse measures the blood flow through the extremities of an individual and can, for example, be captured with a photoplethysmography (PPG) [Mandryk, 2008]. When the sympathetic nervous system increases its activity, for instance because of stress, the blood volume flow in specific parts of the body decreases [Andreassi, 2007]. In research, the blood volume pulse was predominantly correlated with various emotions [Kreibig et al., 2007], [Picard et al., 2001].

Additionally, we relied on the heart rate that refers to the number of contractions of the heart each minute, and the heart rate variability that represents the variation in the time interval between two consecutive heart beats. Under normal conditions, the heart performs around 72 contractions per minute [Andreassi, 2007]. Changes in the heart rate are either caused by a decrease of the parasympathetic nervous system (PNS) activity or an increase in the SNS activity [Andreassi, 2007]. In research, both measurements were linked to mental and cognitive load, as well as stress levels [Richter et al., 1998], [Rieger et al., 2014], [Wilson, 1992]. We used common features such as the mean heart rate, the mean and the standard deviation of the time between two heart beats and features that capture the peaks in the BVP signal.

Breathing-related Measurements. We only used one breathing-related measurement: the respiratory rate (RR). The respiratory rate refers to the number of breaths within a specific time period (most often one minute). Under normal resting conditions, the respiratory rate is between 12 and 15 breaths per minute [Cacioppo et al., 2007]. Research has linked this measurement to mental effort, task difficulty and task demand [Veltman and Gaillard, 1998], [Kuznetsov et al., 2011], [Fairclough et al., 2005]. We extracted commonly used features, such as the mean respiration rate or the $\log_{10}$ variance of the respiration signal.

1.3 Research Approach & Study Setup

In our studies with software developers, we investigated the correlations between biometric measurements and perceived difficulty, emotions, progress and quality concerns. Table 1.1 provides an overview of the three exploratory studies we conducted and the participants we recruited for each of the three studies. In total, we were able to collect biometric data from 47 participants while they were working on our study tasks or their own tasks.

Figure 1.2: One of our participants working on one of our study tasks while wearing a headband capturing EEG data and a wristband recording skin- and heart-related measurements. An eye tracker is placed in front of the screen to capture eye-related measurements.

For each of the three exploratory studies we conducted, we followed a similar study setup. We recruited study participants and asked them to work on certain software development tasks. Each study was planned to be minimally invasive

with respect to time, participants' usual work flow and physical invasiveness caused by the biometric sensors. Our studies lasted from slightly more than an hour up to two weeks per participant. While the study participants were working on their tasks, we recorded their biometric data with various sensors and the relevant outcome measures for each study, such as a developer's rating of his/her perceived progress on a task. Figure 1.2 depicts the study setup for one of our study participants working on one of the study tasks. The participant is wearing a headband to capture EEG data and a wristband to record the skin- and heart-related measurements. In front of the screen, an eye tracker is placed to capture the eye-related measurements. Both the biometric data and the outcome measures were stored for our post-hoc analyses. In addition, we asked study participants to watch a calming two-minute video of fish swimming in a fish tank. The movie helped participants to relax and to record a biometric baseline for each of them.

An overview of our data recording and analysis steps is presented in Figure 1.3. After recording all the data and applying several data cleaning techniques, we extracted commonly used features from the biometric data that research in psychology has already linked to the outcome measures we tried to predict for the specific study. Finally, we fed the extracted features into a machine learning algorithm, trained the classifier, and tried to predict the outcome measure. Each of these steps is described in more detail in the following sections.

Data Recording. During each study, we recorded the outcome measure of interest and certain biometric data of developers. For instance, to collect developers' perceived progress on a task, we periodically interrupted study participants and let them rate their perceived progress on a 5-point Likert scale, ranging from 1 ("completely stuck / no progress at all") to 5 ("in flow / a lot of progress").

To record the biometric data, we used seven different sensors during the course of our studies: a Tobii TX300 eye tracker (see Figure 1.5), an Eyetribe eye tracker (see Figure 1.8), an Affectiva Q Sensor 2.0 (see Figure 1.7) to record EDA data, an Empatica E3 and E4 wristband (see Figure 1.4) to record skin- and heart-related measurements, a Neurosky Mindband (see Figure 1.6) to capture

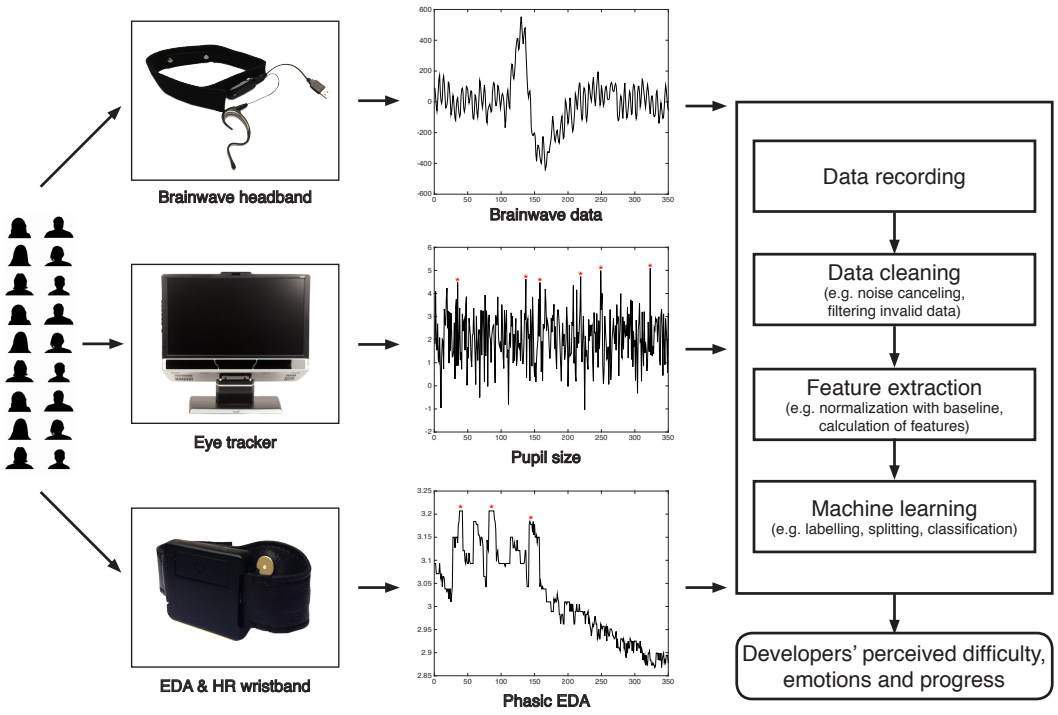


Figure 1.3: Overview of the study method we followed to record biometric data and use machine learning to find measures for a developer's perceived difficulty, emotions and progress.

an EEG, and finally a SenseCore chest band[1] (see Figure 1.9) to record various skin-, heart- and breathing-related measurements. We chose these particular sensors because i) research has shown that the measurements captured with these sensors correlate with the outcome measures we aimed to predict, and ii) these sensors are less invasive than other similar devices.

Data Cleaning. Since biometric data is notoriously noisy, the first step is to clean the captured data. We applied various noise cleaning and filtering techniques depending on the collected data. For instance, to remove noise in the EDA signal, we applied an exponential smoothing filter and we also subtracted the EDA signal's DC component so that the signal is always based at 0 µS. As

[1]Unfortunately, the SenseCore sensor is no longer available.

Figure 1.4: Empatica E4 wrist band [Empatica, 2015].

Figure 1.5: Tobii TX300 eye tracker [Tobii, 2015].

Figure 1.6: Neurosky Mindband [Neurosky, 2015].

Figure 1.7: Affectiva Q Sensor [Affectiva, 2015].

Figure 1.8: Eyetribe eye tracker [Eyetribe, 2015].

Figure 1.9: SenseCore chest band.[1]

another example, for the eye tracking data, we deleted invalid data points as indicated by the eye tracker output.

During the data cleaning step, we also segmented the data as needed for later analyses. For example, to predict a developer's emotions and progress while working on a change task, we only used ten second time windows of biometric data collected just before each time we interrupted developers and asked them to rate their emotions and progress.

Feature Extraction. In many cases, the raw (and cleaned) data by itself is not meaningful and specific features have to be extracted. We therefore extracted features from the biometric data that research has previously linked to outcome measures closely related to what we aimed to predict in our work. Since biometric data can vary significantly between individuals, we also normalized the data per

participant using a baseline. To capture baseline data, we periodically asked the study participants to watch a calming and relaxing two-minute video of fish swimming in a fish tank. We noticed that a participant's biometric features drop down to a baseline after about a minute. Therefore we used the biometric data captured during the second minute of watching the video as a baseline for our normalization approach. A more detailed discussion of the biometric features we extracted can be found in Chapter 1.2.3.

Machine Learning. For training and testing a machine learning classifier, we first labelled and split the data. For the data labeling, we assigned the outcome measure we aimed to predict to the biometric data we used for the prediction. The splitting of the data into training and test data varied across our studies and the evaluation method (*e.g.*, cross-validation or leave-one-out) and required various levels of sophistication. Since our feature sets often also contained different features that are correlated with each other, we performed a feature selection step using manual analysis or algorithms such as `ConsistencySubsetEval` [Liu and Setiono, 1996]. Finally, we chose an appropriate machine learning classifier for each study, trained the classifier and then performed the prediction. We decided to use machine learning since it allows us to predict developers' cognitive and emotional states on the fly, and has been found to be a good approach to identify links between low-level data (*i.e.*, biometric data) and high-level phenomena (*i.e.*, our outcome measure) [Bednarik et al., 2012].

1.4 Findings

In the following, we briefly summarize the main findings of our studies. These findings are presented in more detail in Chapters 2 - 4. We split the presentation of our main findings based on the research questions.

RQ1: Developers' Perceived Difficulty. In our studies, we found initial evidence that biometric measurements can be used to predict developers' perceived difficulty while working on small code comprehension tasks. In particular, we were able to train a Naïve Bayes classifier on biometric data and predict perceived task difficulty for a new developer that we did not train on with 65% precision

and 65% recall. The precision and recall went up to 84%, respectively 70%, when we predicted the difficulty of a new task and trained the classifier on other tasks for each developer. So while these classifiers can be used to predict difficulty even for people they were never trained on, they can be a lot better in predicting when they are trained on the individual they are used for, since biometric data varies a lot across people. In a second study, we further found that biometric data can also be used to predict perceived difficulty in the field, when developers are working on their real change tasks in their usual work environment.

Overall, the findings from our studies support our hypothesis and show that we can use biometric data to predict perceived difficulty (RQ1). More details can be found in Chapters 2 and 3.

RQ2: Code Quality Concerns. The results of our longitudinal study show that biometrics can be successfully used to provide value and predict quality concerns even in the field and over a long period of time. In particular, the results show that code elements that are perceived as being more difficult also tend to contain more quality concerns than code elements perceived as being easy and that biometric data can be used to detect many of the code quality concerns that were later found in peer code reviews. Our results show that our classifier based on biometric data outperforms classifiers based on more traditional metrics and improves upon a naive classifier by more than 26%.

Overall, our findings confirm that we can use biometric data to identify code quality concerns online (RQ2). Chapter 3 presents more details on these findings.

RQ3: Developers' Emotions & Progress. The results from our lab study show that developers' emotions are positively correlated with perceived progress and that biometrics can be used to distinguish between positive and negative emotions in 71% of all cases and between phases of low and high progress in 68% of all cases. The analysis also showed that there is a big variance between individuals and in how accurately one can predict emotions and progress using biometric data, ranging from as little as 30% for one participant to getting all cases right (100%) for another one.

Overall, the results of this study provide evidence of the value of biometrics to determine a developer's perceived progress and emotions while working on a change task (RQ3). More details can be found in Chapter 4.

In summary, the findings from our three exploratory studies support our hypothesis and demonstrate the high potential that biometric data has to measure certain aspects of a developer's work in real-time and with off-the-shelf biometric sensors. In particular, the results show that biometric data, even captured in the field and only using short time windows, allows to accurately predict the difficulty developers experience, as well as their progress and emotions. Additionally, we were able to use biometrics to predict code quality concerns online. These findings open up many opportunities for better supporting developers in their work, for instance by automatically and instantaneously detecting potential quality concerns in the code, or by avoiding costly interruptions when a developer is in the flow and making a lot of progress.

1.5 Threats to Validity

There are several threats to the validity of our research, in particular the external, internal and construct validity.

External Validity. There are several aspects that might limit the generalizability of our findings to other populations or environments.

First, each study presented in this work has a relatively small sample size, ranging from 15 to 17 study participants. We tried to mitigate this risk by selecting participants with varying backgrounds and experience, *e.g.*, students and professional developers. Furthermore, we generally recruited professional developers for our studies from different companies and for our study on predicting developers' perceived difficulty, we replicated our initial study in a different company (see Chapter 3).

Second, the environment in which we conducted a study might threaten the generalizability of our findings. In particular, the findings from our lab studies might not generalize to less controlled environments. We mitigated this risk

by running a combination of studies ranging from very controlled lab studies to field studies in which professional developers were working on their usual change tasks in their normal work environment. In one case, we also confirmed our findings from a lab study in a later on field study. In addition to the work environment, we also carefully chose all study tasks (for the lab studies) to ensure their representativeness of typical change tasks and for our field study, we confirmed the study period's representativeness of the usual work with study participants.

Finally, for most of our studies, we focused on a specific programming language such as Java or C#, or on a specific integrated development environment (IDE), such as Eclipse. This might limit the generalizability of our findings. In general, we tried to mitigate this risk by focusing on popular and common choices, such as Java and Eclipse. Due to the broad variety in the way developers, teams and companies develop software, for instance, with respect to the programming language, change task size, or the software development process used, further studies are needed to investigate the generalizability of our findings.

Internal Validity. The internal validity refers to the degree of confounding factors that might influence the validity of the obtained results. To collect outcome measures, such as perceived progress and difficulty, we regularly interrupted participants in our study and asked them to assess themselves. These interruptions might have influenced the study participants' work. We tried to mitigate this risk by using mediated interruptions and allowing participants to postpone their assessment as well as we chose a time interval between interruptions that is representative of the one for task switches of professional developers.

Construct Validity. The construct validity refers to the question of whether our study designs are appropriate to measure what they claim to be measuring. The biggest threat to the construct validity is the fact that biometric measurements might be influenced by other aspects, such as individual differences or noise, than the ones we tried to measure in our studies. Research has shown that biometric measurements can be affected by the season, the weather, the time of day, or a person's personality traits [Cacioppo et al., 2007]. To mitigate this risk, we applied noise canceling, baseline and normalization techniques to the

captured biometric data. Furthermore, research has also shown that biometric measurements can exhibit a high variability between different humans [Mandryk, 2008]. Again, to account for these individual differences, we applied baseline and normalization techniques before we analyzed the data. Finally, biometric sensors are prone to errors, noise and interferences [Mandryk, 2008]. To account for that, we applied several cleaning and noise reduction techniques.

The way we collected the outcome measures, such as perceived progress and difficulty, also poses a threat to the construct validity. To collect these measures, we regularly asked participants to assess themselves. Their self assessments are subjective and might not be representative of their actual perceptions or the objective progress on a task. Since we focused on predicting the *perceived* cognitive and emotional states, as well as the *perceived* progress on a task, instead of the actual progress, the impact of this threat is limited.

1.6 Challenges

Using biometric sensors and data poses several challenges, in particular due to sensor limitations, privacy and ethical concerns, the difficulty of recruiting study participants, and the analysis of big data.

Sensor Limitations. To collect biometric data, study participants have to wear biometric sensors. Especially due to the invasiveness of some of the biometric sensors, collecting the needed biometric data over longer periods of time can become challenging. In our studies, we therefore chose sensors that are minimally invasive and yet provide the required accuracy and granularity of the biometric data. In the near future, the advances in sensor technology will most likely significantly decrease the invasiveness of biometric sensors. Recent devices, such as the Apple Watch [AppleWatch, 2015] or the Microsoft Band [MicrosoftBand, 2015], are already able to record some biometric data and are much like traditional wristwatches, without being any more invasive.

In addition to the invasiveness, many biometric sensors, including some of the ones used in our studies, are still relatively new and immature, including their data transfer. This poses several challenges to collect the relevant biometric

data. For our studies, we therefore made sure to always have a researcher on-site to help with technical support. Especially for our longitudinal study, we also ensured to always only have small groups of professional developers participate at the same time. With the fast growth of the market and technology, this situation should improve soon.

Privacy and Ethical Concerns. The capturing of huge amounts of fine-grained personal health-related data poses several ethical and privacy concerns. In our studies we made sure to store biometric data only in an anonymized form and provide each study participant with access to only his/her own biometric data. We also strictly limited the access to the captured data to only the researchers directly involved in the studies. However, future research is needed to examine how privacy and security can be ensured when biometric data has to be shared with a broader audience, for instance to provide effective tool support.

Recruiting Study Participants. Due to the sensor limitations and the privacy and ethical concerns, finding participants who are willing to wear these sensors for weeks to capture very personal and sensitive data, was a tedious and time-consuming task. By widely advertising our studies, providing participants with access to their biometric data during the course of the study and choosing minimally invasive sensors, we were able to interest and recruit study participants. In the near future, recruiting study participants might become easier with more mature and less invasive biometric sensors.

Analyzing Big Data. Over the course of our studies, we captured close to 22 GB worth of biometric data, consisting of more than 200 million data points. Much more data was captured for testing, pilot studies and informal experiments. The sheer amount of data poses several challenges with respect to the time and computational power it takes to run the algorithms for noise canceling, normalization and feature extraction. We tackled this challenge by parallelizing the calculations when possible and by running the feature extraction algorithms on a dedicated machine. It should be possible to further speed up the analysis of biometric data by optimizing the data collection step, for instance by reducing the sample rate of the biometric sensors, or by focusing on a subset of biometric features.

Biometric Feature Selection. Finally, the selection of appropriate features for machine learning classifiers poses a challenge as well. As Table 1.2 indicates, research has conducted a myriad of studies on the correlation between biometric measurements and psychological states. For almost all biometric measures there are studies that found correlations between a measure and a psychological state, while other studies could not find a significant correlation. Therefore, it is important to carefully choose the features to be used. In our research, we made sure to focus on biometric measurements for which most relevant studies have found a correlation and also preferred features for which a correlation has been found in studies with a similar study setup. However, future research is needed to further investigate which biometric features to prefer in which context.

1.7 Opportunities & Future Work

Overall, the results of our research studies demonstrate the high potential that biometric data has to measure certain aspects of a developer's work in real-time and with low-cost, off-the-shelf biometric sensors. In particular, the results show that biometric data can be used to predict the difficulty developers experience, their progress and code quality concerns with high accuracy. These findings open up new opportunities for providing better developer support and for increasing every individual developer's productivity.

Measuring Individual Developers in Real-Time. A drawback of traditional approaches to assess (task) difficulty, code quality and a developer's productivity is that they do not take into account the individual differences that exist between developers, such as their experience, their mood, or their general well-being. All these aspects might influence developers' perceived difficulty on a change task and might also affect their output. Biometrics shift the focus towards the individual developer and take into account their differences. While this requires some training of machine learning classifiers, our studies have shown that using biometrics can outperform traditional metrics and provide more accurate predictions.

Another drawback of traditional approaches is the fact that they rely on metrics that are often only available after a developer completed a change task and thus only allow for a post-hoc analysis. Biometric data on the other hand has the potential for real-time measures. Results of our studies already show that a very small amount of biometric data (*i.e.*, seconds worth of biometric data) is sufficient to accurately predict perceived difficulty, emotions, progress and quality concerns. Also, current biometric sensors already provide the possibility to stream biometric data in real-time, but more research is needed to investigate the possibility of full real-time support.

Developer Support. Biometrics, in particular their potential to measure developers' experiences in real-time, open up new possibilities for supporting developers. For instance, a real-time measure of the difficulty a developer experiences while working on a specific code element might be used to automatically detect quality concerns in the code or when a developer is likely to create a bug in the code. This might allow us to prevent developers from submitting bugs to the code repository, or at least help to identify places in the code that would particularly benefit from a thorough code review. With the recent advances in eye tracking technology, it might then even be possible to identify code quality concerns not only on method or class level, but even on line or statement level.

Finally, each person has times during the day or even days when they are more or less focused and productive. Several researchers in psychology, for instance, distinguish between morning and evening people [Levandovski et al., 2013]. Measures that allow us to capture the cognitive and emotional states of developers, will allow us to provide better support for each developer. For instance, by knowing when a developer is most productive during the day, we might be able to optimize the work day and schedule the most demanding tasks for these times or avoid costly interruptions. We would also be able to provide more tailored and in-time recommendations, such as suggesting to talk to a coworker for help or taking a break when a developer is stuck and frustrated.

Predicting Further Psychological States. The goal of our research is to measure specific aspects of a developer's work. In particular, we have focused on developers' perceived difficulties, emotions and progress. However, there are

many other aspects that influence a developer's work and that biometrics might be able to capture.

One example is a developer's interruptibility — how disruptive an interruption is to a developer at a given point in time. Developers frequently face interruptions during their work, *e.g.*, from coworkers asking a question in person, instant messages or emails. Although some of these interruptions are beneficial and wanted, many interruptions occur at inopportune moments and can slow down developers significantly as well as cause a higher error rate [Bailey and Konstan, 2006], [Czerwinski et al., 2004]. Biometrics might allow us to determine when a developer is more or less interruptible during a work day [Züger and Fritz, 2015] and thus help to manage interruptions better. Concerning developers' interruptibility, we have already conducted a field study and our initial results show the potential that biometrics have in this scenario. Similar scenarios are imaginable if we are able to use biometrics to determine a developer's task engagement.

Smart Wearable Devices. The term 'Smart Wearable Devices' refers to electronic devices that are integrated in clothes or accessories and provide their users with some kind of real-time feedback about activities or physiological features. The market for smart wearable devices is growing at an immense speed. From a market value of 600 million US$ in 2013, the market has increased to around 4 billion US$ in 2014 and will, according to forecasts, reach 30 billion US$ in 2020 [Hunn, 2014]. These devices come in many different forms, such as smart glasses like the Google Glass [GoogleGlass, 2015], smart watches like the Apple Watch [AppleWatch, 2015] or fitness trackers like the Nike+ Fuelband [Nike+ Fuelband, 2015] or Fitbit [Fitbit, 2015].

The biometric sensors that are integrated in some of these devices are becoming more and more sophisticated and will soon support the capturing of biometric data that might be fine-grained and accurate enough for some of the presented scenarios. With the fast growing market and pervasiveness of smart wearable devices, an increasing amount of people will be wearing such devices. This will provide the opportunity to capture and collect biometric data for many developers on a daily basis and allow for the fine-tuning of classifiers on aspects

such as task difficulty or progress which will further increase the potential the data has.

1.8 Related Work

Work related to our research can be categorized into four major areas based on the kind of research we perform and the three research questions we address: 1) research on the use of biometrics in the software development context, 2) studies investigating how (task) difficulty can be measured, 3) research on identifying code quality concerns, and finally 4) research on developers' emotions and progress.

1.8.1 Biometrics in Software Engineering

In the software development domain, biometric sensors have been and are predominantly used to gain a better understanding of developers' program comprehension, using either eye tracking or by measuring brain activities. Only a few approaches have used other sensors and/or looked at other aspects, such as task difficulty.

Early on, Crosby *et al.* [Crosby and Stelovsky, 1990] studied the differences in program comprehension between experienced and less experienced software developers using eye tracking technology. More recently, Bednarik *et al.* [Bednarik and Tukiainen, 2006], [Bednarik and Tukiainen, 2008] conducted similar studies to investigate different strategies of novices and experts in program comprehension and debugging tasks. In a similar direction, Sharif *et al.* [Sharif and Maletic, 2010a] used eye tracking for small program comprehension tasks to investigate the effect of naming conventions — *camelCase* and *under_score* — on the success of solving the program comprehension tasks. Finally, Sharafi *et al.* [Sharafi et al., 2012] relied on eye tracking technology to investigate the differences in reading strategies of male and female participants when working on program comprehension tasks.

Using brain activity measures, Ikutani *et al.* and Siegmund *et al.* studied developers during small program comprehension tasks. Ikutani *et al.* [Ikutani and Uwano, 2014] used near-infrared spectroscopy to investigate the differences

in brain activity for various types of program comprehension tasks. Siegmund *et al.* [Siegmund et al., 2014] used functional magnetic resonance imaging to identify active brain regions during program comprehension. The results of their study show that brain regions that are related to working memory, attention and language processing are active during program comprehension.

One of the early approaches to mention multiple biometric sensors to study software developers was the Ginger2 environment by Torii *et al.* [Torii et al., 1999]. In their approach they included an eye tracker and a skin resistance level sensor into an environment that was built to continuously collect a variety of data from software developers participating in empirical studies. More recently, with the advances in sensor technology and the availability of affordable biometric sensors, a few researchers have also looked at other aspects, such as task difficulty. For instance, there is some preliminary work on the use of biometric measurements to assess developers' mental load and perceived difficulty while working on small code snippets. Parnin [Parnin, 2011] investigated the use of electromyography to measure sub-vocal utterances and found that these measurements might be used to assess programming task difficulty. Similarly, Nakagawa *et al.* [Nakagawa et al., 2014] were able to use cerebral blood flow (CBF) to distinguish between two difficulty levels while developers were performing code comprehension tasks.

In our research, we extend these studies by (a) using a combination of various biometric sensors (b) for multiple outcome measures, including task difficulty but also code difficulty and emotions and progress, and (c) by conducting several studies in various contexts, including a field study, with more realistic tasks and more participants.

1.8.2 Assessing (Task) Difficulty

Research on assessing task difficulty can be divided into empirical studies and approaches for automatic assessment. Empirical studies have been conducted to investigate the common difficulties that developers faces during code comprehension tasks [Karahasanović et al., 2007], [Tiarks and Roehm, 2012]. To automatically assess difficulties that developers experience, researchers have mainly focused on the use of metrics of the artifacts that developers work with.

For instance, Kasto *et al.* and Katzmarski *et al.* used size and complexity metrics to assess the difficulty of code comprehension tasks and found that for some study participants and some tasks, the code metrics can be an indicator of task difficulty [Kasto and Whalley, 2013], [Katzmarski and Koschke, 2012].

All these approaches suffer from two major drawbacks: first, the used code metrics can most often only be collected after the work on a change task is completed, and second, they do not take into account the individual differences that exist between software developers. In our research, we focus on the use of biometrics to automatically and instantaneously predict developers' difficulties. By using biometric measurements, we are also able to take the differences between individual software developers into account.

1.8.3 Identifying Code Quality Concerns

Related work concerning the identification of code quality concerns can be further split into manual and automatic approaches. Several studies have shown that code reviews can lead to a significant improvement of code and reliable detection of defects soon after code changes were performed [Bacchelli and Bird, 2013], [Bosu et al., 2015], [Rigby et al., 2008]. However, manual approaches require much developer time and effort.

To speed up the detection of quality concerns, much research has looked into the automatic prediction of defects. Most of this research focuses on various code metrics, such as complexity, size or change metrics to identify software defects [Gray et al., 2009], [Nagappan and Ball, 2005], [Nagappan et al., 2006], [Zhang et al., 2007]. Other researchers have also looked into organizational information [Nagappan et al., 2008], [Weyuker et al., 2008] and change history [Palomba et al., 2013]. Almost all of these approaches do not take into account the individual differences between developers, and can only be used for a post-hoc detection. A first step into the direction of an individual, online identification of code quality concerns was taken by Lee *et al.* [Lee et al., 2011] who investigated developers' individual interaction patterns within the IDE and assessed how these measurements can be used to improve defect prediction.

In contrast to all these studies, we use biometrics to identify code quality concerns. By using biometrics, we are able to take into account the individual differences between developers and detect defects online, *i.e.*, while a developer is still working on the change task.

1.8.4 Developers' Emotions & Progress

So far, only a few studies have investigated software developers' emotions and how these emotions can affect their perceived progress and productivity. One of the first studies in this direction was conducted by Shaw [Shaw, 2004] who observed students working on a software project and found that their self-reported emotions can change significantly within 48 hours. In a similar direction, Wrobel *et al.* [Wrobel, 2013] conducted a survey to investigate how emotions impact software developers' effectiveness at work. Besides other findings, they found that for some people, negative emotions can have a positive effect on their productivity. The correlation between emotions and productivity was also investigated by Graziotin *et al.* [Graziotin et al., 2013] who examined whether valence, arousal and dominance correlate with the self-reported performance of software developers. They found that valence and dominance are positively correlated with a developer's performance.

Different to these studies, Khan *et al.* [Khan et al., 2011] actively induced moods and measured their impact on developers' debugging performance. Furthermore, in two other studies, and closer to our research, Khan *et al.* [Khan et al., 2013] investigated how moods can be measured with keyboard and mouse input and found individual correlations between mood and keyboard/mouse input, but no generic correlation valid for all study participants. In our research, we extend results from these studies by providing additional evidence of the correlation between emotions and progress and further investigate how biometric data can be used to predict developers' emotions during change tasks.

Researchers have also looked more specifically at the progress developers make and determining it automatically. For instance, Carter *et al.* [Carter and Dewan, 2010b] mined IDE interaction logs to automatically determine when a developer is stuck and cannot make any progress. Closer to our research, Muldner *et*

al. [Muldner et al., 2010] used a combination of posture, skin conductance, and pressure sensors, as well as an eye tracker, to determine instances when a student made progress while solving physics exercises. In contrast to these studies, we use biometrics to predict both emotions and perceived progress, and focus on realistic software change tasks.

1.9 Summary & Contributions

The initial results from our three exploratory studies confirm our hypothesis that a combination of biometric sensors can be used to determine a software developer's perceived difficulty, emotions and progress when working on a change task, as well as to predict potential code quality concerns in code elements. In general, we have shown that biometric measurements have a great potential to determine many different emotional and cognitive states of a developer working on change tasks. Our work makes the following contributions:

- it presents results from a lab and a field study that show that biometric data can be used to assess a developer's perceived difficulty with high accuracy;
- it presents evidence from a field study that shows that classifiers based on biometric data are able to identify code quality concerns while a developer is working on the code, also outperforming classifiers based on more traditional metrics;
- it presents results from a lab study on the viability to use biometric measures to classify developers' emotions and perceived progress with high accuracy during a software development change task;
- it discusses the use of biometrics in the context of software engineering and summarizes research approaches, findings, future opportunities and challenges; and
- it introduces and presents a reusable framework for recording, cleaning and analyzing biometric measurements.

This is a first step towards a better understanding of a developer's perceptions during programming, and in the long term, towards increasing every individual developer's productivity. Our findings open up a many opportunities to boost a developer's productivity, for instance by automatically and instantaneously detecting code quality concerns, or by reducing costly interruptions when a developer is in the flow and making a lot of progress on a task. With the fast advances in sensor technology and the increased proliferation of smart wearable devices, it might soon be possible to improve the precision of these kind of predictions and capture data in less invasive ways. Despite the promising results, there are also several challenges and threats to the validity that have to be addressed in the future, such as ethical and privacy concerns and the generizability of our initial results.

1.10 Thesis Roadmap

The remainder of this thesis consists of five chapters. Each chapter is based on a scientific publication as illustrated in Figure 1.10. All five publications were published at internationally renowned, peer-reviewed conferences. Three of these five publications (Chapter 3, 4 and 6) were created in collaboration with my supervisor, Prof. Thomas Fritz. For the publication presented in Chapter 2, we worked together with Andrew Begel from Microsoft Research, Manuela Züger from the University of Zürich and Serap Yigit-Elliot. Finally, the publication presented in Chapter 5 was created in collaboration with Timothy R. Shaffer, Jenna L. Wise, Braden M. Walters, Michael Falcone, and Bonita Sharif, all from the Youngstown State University.

Chapter 2 presents the results of a lab study on the use of biometric measurements to predict whether small code comprehension tasks are difficult or easy for an individual developer. This study presents an initial step to investigate the use of biometric sensors in the context of software development.

Chapter 3 presents the results of a two-week field study that investigates the use of biometrics to determine code quality concerns and the difficulties developers

Figure 1.10: Thesis Roadmap. Highlighted publications indicate first-authored papers.

experience on code elements in a real world setting. This study builds on and extends our initial lab study.

Chapter 4 reports on the results of a lab study that examines the use of biometric sensors for assessing emotions and progress that software developers experience while working on a change task.

Chapter 5 presents iTrace, an Eclipse plugin that is able to record a developer's eye movements and further eye-related measures and match them to the locations in the code the developer is looking at. This plugin contributes to our framework for using biometrics, in this case an eye tracker, in the software development domain.

Chapter 6 presents an overview of the field of biometric sensing in the software development domain and summarizes research approaches, findings, future opportunities and challenges.

2

Using Psychophysiological Measures to Assess Task Difficulty in Software Development

Thomas Fritz, Andrew Begel, ***Sebastian C. Müller****,*
Serap Yigit-Elliott, and Manuela Züger
Published at the 36th International Conference on Software Engineering, 2014
Contribution: Data analysis and paper writing

Abstract

Software developers make programming mistakes that cause serious bugs for their customers. Existing work to detect problematic software focuses mainly on *post hoc* identification of correlations between bug fixes and code. We propose a

new approach to address this problem — detect when software developers are experiencing difficulty while they work on their programming tasks, and stop them before they can introduce bugs into the code.

In this paper, we investigate a novel approach to classify the difficulty of code comprehension tasks using data from psychophysiological sensors. We present the results of a study we conducted with 15 professional programmers to see how well an eye-tracker, an electrodermal activity sensor, and an electroencephalography sensor could be used to predict whether developers would find a task to be difficult. We can predict nominal task difficulty (easy/difficult) for a new developer with 64.99% precision and 64.58% recall, and for a new task with 84.38% precision and 69.79% recall. We can improve the Naive Bayes classifier's performance if we trained it on just the eye-tracking data over the entire dataset, or by using a sliding window data collection schema with a 55 second time window. Our work brings the community closer to a viable and reliable measure of task difficulty that could power the next generation of programming support tools.

2.1 Introduction

Knowing how hard a task is as it is being performed can help in many dimensions. For instance, the estimate for completing a task might be revised or the likelihood of a bug occurring in the source code changes for the task might be predicted. Existing work to determine task difficulty has mainly focused on already existing artifacts, such as task descriptions, and the similarity of artifacts using machine learning classifiers. In our research, we are investigating a novel approach to determine task difficulty that uses psychophysiological data gathered from the developer while he is working, such as electroencephalographic (EEG) activity along the forehead or electrodermal activity (EDA). By using psychophysiological sensors and collecting data while a developer is performing a task, we present the first approach that can support an instantaneous measure of task difficulty that does not rely on already produced artifacts or even whether the developer is writing any code at all.

There has been extensive research in psychophysiology investigating how various measures can be linked to psychological states and processes (*e.g.*, [Kramer, 1991], [Rowe et al., 1998]), but only little work has investigated the use of such measures in software development. Predominantly this work used eye-tracking technology to retrospectively determine the effect on visual effort for different representations, such as differences in identifier styles or the visual representation of requirements (*e.g.*, [Sharif and Maletic, 2010a], [Sharafi et al., 2013]). None of this work has used psychophysiological features of software developers to measure task difficulty. One preliminary pilot study by Parnin [Parnin, 2011] has explored the use of electromyography to measure sub-vocal utterances and investigate them as an additional measure for task difficulty. While Parnin found a correlation between utterances and a developer editing code, his work looks at only a single psychophysiological feature, ignoring the potential to look for instantaneous, or more general measures of psychophysiological features corresponding to task difficulty.

In this paper, we investigate whether we can use psychophysiological measurements to determine whether a code comprehension task is perceived as easy or difficult. In particular, we ask the following questions:

(RQ1) Can we acquire psychophysiological measures from eye-tracking, EDA and EEG sensors to accurately predict whether a task is difficult or easy?

(RQ2) Which combination of psychophysiological sensors and associated features best predicts task difficulty?

(RQ3) Can we use psychophysiological measures to predict task difficulty as the developer is working?

With such a code- and quality-independent indicator for a developer's difficulty with a task, it may be possible to design a set of interventions that could prevent the developer from introducing bugs caused by cognitive difficulties, and also provide timely support for the remainder of his task.

To answer our research questions, we conducted an exploratory study in which 15 professional software developers monitored with psychophysiological sensors

performed six to eight tasks. We gave the developers code comprehension tasks that were small, but large enough to challenge the subjects for a few minutes at a time. Using all of the sensor data, we were able to train a classifier to predict whether a developer, on which the classifier was not trained on, would perceive the task to be easy or difficult with 64.99% precision and 64.58% recall. Using just the eye-tracking data resulted in even greater predictive power. To create a classifier that can operate while the developer does his work, we explored how well a sliding time window data collection approach (adjusting the size of the time window from 5 second to 60 seconds, sliding it 5 seconds each time) could make predictions of the developer's final assessment of task difficulty after finishing his task. We found that combining subsets of sensors with particular time windows could improve classifier performance when predicting a new developer's task difficulty and a developer-task task difficulty pair.

Our contributions are

- an exploratory study on the viability of using psychophysiological sensors to determine code comprehension task difficulty;
- an approach to classify tasks by difficulty using time intervals suitable for on-the-fly classification;
- and an investigation of which combination of psychophysiological sensors and measurements are most effective at predicting task difficulty.

Overall, our work provides the software engineering research field with a new perspective on using psychophysiological measures to understand and support the software developer in his activities. In the future, advances in sensor technology and data analysis techniques should make it possible to employ simpler, cheaper, and more accurate metrics and develop them into programming support tools.

2.2 Related Work

Related work can be categorized into two areas: the general use of psychophysiological measures to study psychological states and processes and research related to aspects of software development using psychophysiological measures.

2.2.1 Using Psychophysiological Measures

There is a broad range of psychophysiological measures that have been explored and linked to psychological, and specifically cognitive, processes and states. All of these measures have different strengths and weaknesses with respect to aspects such as invasiveness, sensitivity, generalizability, interpretability and ease of collecting [Kramer, 1991], [Rowe et al., 1998]. Some of the most commonly used measures can roughly be categorized into eye-related, brain-related or skin-related measures.

Eye. There is a variety of eye-related measures, such as the pupil size, fixation duration or number of saccades. Early on, Beatty found that task-evoked pupillary response, in particular the peak amplitude of the pupil diameter, is an indicator for memory load or also processing load and that it varies with task difficulty [Beatty, 1982]. Further research on pupil size found similar correlations, *e.g.*, to mental workload of subtasks [Iqbal et al., 2004] and cognitive load [Klingner, 2010], and even used pupil dilation as a measure for workload at task boundaries [Bailey and Iqbal, 2008]. Others used measures of fixation and saccades, *e.g.*, Goldberg *et al.* found that a higher number of saccades is an indicator for a poorer interface [Goldberg and Kotval, 1999], and in their overview on eye-tracking research in HCI and usability, Jacob *et al.* state that the mean fixation duration is believed to be an indicator of a participant's difficulty in extracting information from a display [Jacob and Karn, 2003]. Recent approaches have also used eye-related measures to train machine learning classifiers and predict a person's cognitive state (*e.g.*, [Simola et al., 2008], [Eivazi and Bednarik, 2011]).

While measures on pupil size, fixations and saccades are commonly captured using an eye-tracking sensor, eye-blink rate is better measured through electrodes placed around the eye, *i.e.*, electrooculography (EOG) or by filtering certain frequencies within electroencephalography (EEG). Studies on eye-blink rate have shown that it is inversely correlated with attention or mental load, *i.e.*, the lower the blink rate, the higher the mental load or attention (*e.g.*, [Telford and Thompson, 1933], [Holland and Tarlow, 1972], [Holland and Tarlow, 1975], [Bauer et al., 1985]).

Brain. With brain-related measures we refer to the recording of electrical activity inside the brain or close to the surface of the scalp, *i.e.*, electroencephalography (EEG). Studies have shown that specific frequency bands, often referred to as alpha, beta, gamma, delta and theta, within the EEG data can be connected to different mental states [Berger, 1929]. For instance, several studies found that a decrease of alpha EEG activity and often an increase in theta EEG activity was accompanied with an increase in attentional demand and working memory load (*e.g.*, [Sterman et al., 1993], [Gevins et al., 1998], [Smith and Gevins, 2005]). Other studies examined an EEG task engagement index defined as "beta / (alpha + theta)" (*e.g.*, [Kramer, 1991], [Lawrence J. Prinzel et al., 2001], [Berka et al., 2007]) based on evidence that with increases in task engagement, theta is suppressed, alpha is blocked and beta increases in relative power, or they found that the theta and delta band are sensitive to task difficulty manipulations (*e.g.*, [Brookings et al., 1996]). As with eye-tracking measures, researchers have also investigated using EEG data and machine learning to predict aspects, such as the working memory load or the cognitive task (*e.g.*, [Lee and Tan, 2006], [Grimes et al., 2008]).

Skin. Electrodermal activity (EDA), also known as skin conductance (SC) or galvanic skin response (GSR), has been closely linked with arousal, attention, emotional states, stress and anxiety [Boucsein, 2012], [Dawson et al., 2007]. Frequently, features of electrodermal activity have been used in combination with measures such as blood-volume pressure and respiration to classify the data into classes or states of emotion (*e.g.*, [Picard et al., 2001], [Maaoui and Pruski, 2010]). In addition, studies have shown that EDA measures can be used to indicate cognitive load levels, task difficulty level and distinguish cognitive load at the workplace from stress (*e.g.*, [Shi et al., 2007], [Nourbakhsh et al., 2012], [Setz et al., 2010]). For instance, Nourbakhsh *et al.* have shown that normalized frequency domains of electrodermal activity were significant to indicate task difficulty levels for arithmetic and reading tasks [Nourbakhsh et al., 2012]. Researchers have also investigated EDA as a real time measure, *e.g.*, to adapt the workload of an operator and avoid it to become too high [Haarmann et al., 2009] or to detect emotions and improve the gaming experience [Nakasone et al., 2005].

Finally, researchers have combined various of these measures. Wilson, for instance, measured brain activity, eye blinks, electrodermal activity and heart rate and found that electrodermal activity measures as well as alpha and delta bands of brain activity showed significant changes to varying mental workload demands in flying scenarios, while the heart rate was less sensitive [Wilson, 2002]. Similarly, Ryu *et al.* combined multiple sensors and found that a combination worked well for distinguishing between the difficulty levels of tasks [Ryu and Myung, 2005]. More recently, Haapalainen *et al.* collected data using multiple sensors, including a NeuroSky mindset for EEG, eye-tracking and a GSR armband and compared their ability to assess cognitive load using six elementary cognitive tasks with varying difficulty levels each. Their results show that electrocardiogram median absolute deviation and median heat flux measurements were most accurate to classify between low and high cognitive load [Haapalainen et al., 2010].

Similar to the aforementioned research, we also want to take advantage of the psychophysiological measures and differentiate between difficult and easy tasks. In particular, we are looking into a combination of sensors, similar to the ones by Haapalainen [Haapalainen et al., 2010]. However, we are looking at aspects of software development and thus differ in the tasks and participants we are studying. In particular, the tasks in our study are more closely related to software development tasks and the participants are professional software developers. Since early research on reading algorithms has found differences to reading prose [Crosby and Stelovsky, 1990], our study provides insights on how these psychophysiological measures could be used in the software development domain.

2.2.2 Psychophysiology in SD

A few studies have investigated the use of psychophysiological measures in software development, mainly using eye-tracking. An early study on code comprehension by Crosby *et al.* used eye-tracking to study the scan patterns and strategies of high and low experienced developers. In their study, they used eye fixation as a measure of attention, classified code into 5 categories from easy to hard and found that high experienced developers use less time on comments and

more time on complex statements than low experienced developers [Crosby and Stelovsky, 1990]. More recent studies by Bednarik *et al.* also analyzed differences in strategies for less and more experienced developers in program comprehension and debugging [Bednarik and Tukiainen, 2006], [Bednarik and Tukiainen, 2008].

Using eye-tracking technology, researchers have also studied the effect of different representations in software development on the visual effort. For instance, Sharif *et al.* looked at the effect of identifier naming conventions — camelCase and under_score — in code comprehension and found that the accuracy in answers stays the same but time and visual effort decreases [Sharif and Maletic, 2010a]. While Sharafi *et al.* also looked at memorability of identifier styles they examined the impact of gender on source code reading and found different comprehension strategies in male and female subjects using eye-tracking [Sharafi et al., 2012]. Studies have also looked at differences in other representations, such as graphical and textual representations of code variables [Nevalainen and Sajaniemi, 2005], requirements [Sharafi et al., 2013] as well as the representation and layout of design patterns [Sharif and Maletic, 2010b], [Porras and Guéhéneuc, 2010].

To study the link between code reviews and defect detection, researchers have also examined the time developers spend scanning code by summarizing fixation durations over specific areas of interest and counting the number of fixations. Thereby, they found that a longer scan time correlates significantly with a better defect detection [Uwano et al., 2006], [Sharif et al., 2012].

All of these approaches examine software engineering aspects, however, they are limited to eye-related features. Khan *et al.* have looked into another psychophysiological aspect and its link to performance by investigating how the mood of developers affects debugging and programming [Khan et al., 2011], [Khan et al., 2007]. In these studies, different moods were induced by showing developers video clips and then the developers' performance was measured. None of this research has investigated the use of psychophysiological measures for determining task difficulty. Closest to our work is a preliminary pilot study by Parnin, who has explored the use of electromyography to measure sub-vocal utterances [Parnin, 2011]. From early results, he found that such a measure

might be used to determine the difficulty of a programming task. While these initial results indicate the potential of these measures, this paper goes further in investigating multiple psychophysiological measures and their relation to task difficulty in a study with fifteen professional software developers.

2.3 Experiment

We conducted a lab experiment with 15 professional software developers. Each subject performed eight code comprehension tasks as we recorded various psychophysiological and subjective measurements[1].

2.3.1 Subjects

Subjects were recruited from a pool of professional software developers who lived in the greater Seattle, WA area and had registered their interest in participating in user studies at Microsoft. A screening questionnaire selected 20 of these candidates who had at least 2 years of software development experience, knew how to program in C# (and had done so in the last year), did not need to wear bifocal or trifocal glasses (they interfere with the eye-tracker), and were available to come to our lab. Five of the 20 selected subjects did not show up. Those that completed the 1.5 hour experiment were remunerated with a single license for their choice of Microsoft consumer software (a standard payment for Microsoft user study participants). Fourteen of the subjects were male and one was female. Subjects ranged in age from 27 to 60 years of age (mean 41.6, stdev 8.2).

2.3.2 Data Capture

We recorded study data using three psychophysiological sensors: eye-tracking, EDA, and EEG. We also recorded the subject's think aloud narrative, recorded a video of the experiment, and recorded a screen capture. The subjects filled out a pre-questionnaire, a written NASA TLX survey [Hart and Staveland, 1988] after

[1] A replication package of the experiment is available via
`http://research.microsoft.com/apps/pubs/?id=209878`

completing each experimental task, and a post-questionnaire after the entire experiment that asked them to rank each of the tasks by relative perceived difficulty. The experiment administrator also took hand-written notes.

Eye-tracking has been used to assess task difficulty and cognitive and mental load [Klingner, 2010], [Iqbal et al., 2004], [Ali et al., 2012], [Brookings et al., 1996]. We used a Tobii TX300 eye-tracker using a 300 Hz tracking frequency to collect gaze location information, fixation and saccade count and duration and pupil diameter. The eye-tracker has an accuracy of 0.4° of visual angle, which is equivalent to 13 pixels on its built-in 96 dpi 1920 x 1080 23-inch monitor. We applied Tobii Studio's built-in I-VT fixation filter with default parameters in order to classify eye movements based on the velocity of shifts in the eyes' directions. To avoid gaze inaccuracy, we directed the tool and our subsequent analysis to record and analyze data only from the subject's dominant eye (determined as part of our experimental procedure).

Electrodermal activity (EDA) is an oft-used sensor to detect arousal, particularly cognitively-determined arousal [Boucsein, 2012]. To measure EDA, we used an Affectiva Q Sensor 2.0 [Poh et al., 2010] worn on the wrist of the subject's non-dominant (and non-mouse-holding) hand. The Q Sensor samples at a rate of 8 Hz, simultaneously measuring skin temperature along with three-axis acceleration data. Data is stored on the device itself, and streamed via Bluetooth to a recording computer.

Electroencephalography (EEG) refers to the measurement of the brain's electrical activity that arises from neuronal firing [Andreassi, 2007]. It is used in a variety of fields, such as neurology and Brain-Computer Interface (BCI) research. There are a variety of devices that record multi-channel EEG signals using sensors attached with gel to various points on a subject's head. To make our experiment less invasive (and minimize cleanup), we decided to use an off-the-shelf NeuroSky MindBand EEG sensor. It is a one-channel, noise-canceling, dry sensor that records the EEG signal at 512 Hz from a single location on the subject's forehead (reading signals mainly from the pre-frontal cortex). The MindBand produces a single, pre-filtered, time-varying voltage signal, as well as two computed signals, Attention and Meditation, corresponding to paying

attention and feeling calm and centered [Neurosky, 2009]. While these are both produced by proprietary (read: trade secret) algorithms, the signals are always available from the entire family of NeuroSky sensors.

Audio/video capture of the experiment was done with two 60 fps cameras, one pointed straight at the subject from the screen (like a webcam) and the other above the subject aimed at the screen and keyboard. The Tobii Studio v3.0 software that was used to run the eye-tracker also recorded the full resolution screen at 60 Hz and added a "follow-the-bouncing-ball" visualization on top of the recording to visualize the subject's gaze location.

We attempted to use all of the psychophysiological sensors to record psychophysiological data for all of our tasks. After refining our procedures with a two person pre-pilot, we were able to successfully capture the complete set of sensor signals for 12 of our 15 participants. We got eye-tracking data for everyone, EEG data for 13 out of 15 participants, and EDA data for 12 out of 15 participants. Two participants failed to produce a valid EEG signal; coincidentally, they failed to produce a measurable EDA signal as well. Another participant's EEG data was lost during capture.

2.3.3 Experimental Tasks

Subjects were asked to perform short (several minutes) code comprehension tasks. In two pre-pilots, we had asked subjects to perform more complex 15-30 minute tasks involving code comprehension and mental code execution, but we found it difficult to scale our characterization of our subject's activities to a granularity of tens of milliseconds for such a long period of time. We eventually designed smaller, shorter tasks, though still limited to code comprehension and mental execution. While these are much simpler than the tasks of many software professionals, we believe that this starting point helps us identify the big picture answers to our research questions, and leaves more details to future experiments.

Each subject was asked to work on ten tasks: two practice questions and eight which were measured. During each task, they were asked to read a short passage of C# code presented on a single screen in the Visual Studio 2012 IDE.

Syntax highlighting was enabled, there were no code comments, and they never executed the code.

There were two kinds of programs. The first created two Rectangle objects, assigned the coordinates of the corners, and "drew" them on the screen. A printed question underneath the program asked the subjects whether the two rectangles overlapped (yes or no). The second program created four shape objects (choosing among Circles, Squares, Rectangles, and Triangles), and then "drew" them in some order on the screen. A printed multiple-choice question underneath the program asked subjects to tell us which three shapes were drawn on the screen last, and the order they were drawn in from five possible answers.

We used three instances of the first program in the experiment. The practice version of this program was used solely to familiarize the subject with the task. One experimental instance was identical to the practice version except for the use of local variables of non-mnemonic single letters. In contrast, the other experimental instance randomized and interleaved assignments of the corner coordinates for both rectangles. This program was designed to stress the subjects' abilities in spatial relations (deciding whether the two rectangles overlapped) and visual object grouping (interleaving the Rectangle initialization statements) and working memory (randomizing the order of the assignments prevents chunking each Rectangle's assignment sequence together, and fills up working memory to a greater amount).

There were seven instances of the second program. They differed in

(a) the order between initialization and drawing each shape (*e.g.*, creating a shape and then drawing it, or creating all shapes and then drawing them in randomized order),

(b) the variable names (mnemonic vs generic) to impact subjects' working memory by interfering with their ability to remember the mapping between variable name and its shape,

(c) using an array to group the shapes and then looping over the array,

(d) making the loop construct mathematically more complex to stress the working memory (for remembering the order of shapes) and their mathematical skills,

(e) calling a separate function to swap the order,

(f) including a double-nested question-mark-colon conditional operator to engage the subject's mathematical and working memory abilities (comparing variables to constants).

Each of the tasks was designed to take subjects between 2 and 5 minutes to finish. Subjects could see both the code and the question on screen at the same time, and never needed to scroll. In fact, we directed the subjects to keep their hands still on the table to avoid affecting the EDA sensor through wrist motion.

2.3.4 Experimental Procedure

① When each subject first entered the lab, he was asked to fill out a consent form and a pre-questionnaire requesting demographic information. ② We synced the internal clock of the EDA sensor to the time on the eye-tracking computer and then placed it on the wrist of the subject's non-dominant hand (the hand that does not use the mouse). ③ We then connected the EDA sensor via Bluetooth to the data recording computer and checked the live display to verify that a signal was being received. Since the EDA sensor works by detecting the electrical conductivity across the wrist, it sometimes fails to work if the subject has no sweat. For those few subjects who did not register any signal, we asked them to do a mild physical exertion (jumping jacks and walking up and down a flight of stairs) to cause them to sweat a little. This sufficed for all but three subjects, so we were unable to record their EDA signal.

During our pilot study, we had noticed that the subjects' EDA signal kept rising monotonically as they completed each subsequent task. This would cause an intense "learning" effect on the EDA signal, so we changed our protocol. ④ Prior to the first task, and in between each one, subjects were asked to watch two minutes of one of four different, calming, full screen YouTube videos of fish swimming in a fish tank and were requested to relax their minds. This relaxation caused the subjects' EDA measurements to return to baseline after about a minute. We were then able to use the EDA signal in the second minute of the video as a baseline for the EDA signal in the next task.

⑤ Next we determined the dominant eye of the subject so we could be sure that our subsequent analysis of the eye-tracking gaze location data would point to the actual word that the subject was reading. ⑥ Each subject was asked to sit on non-wheeled chair in front of the eye-tracking computer and shift the seat around until their head stayed in an imaginary box about 50–75cm in front of the center of the screen. No chin rest or mouth guard was required. A peripheral display on a second computer enabled the experimenter to notice if the subject moved too far out of range (> 37 cm side to side and/or > 17 cm up and down) and ask him to move back into range before continuing.

The subjects were then shown the practice tasks in Visual Studio 2012, and the font size was adjusted, if requested by the subject. We asked them to think aloud while doing the task, and to tell us the answer out loud rather than typing it into the computer. We turned on the audio and video recording and helped them put the MindBand EEG sensor on their head. We then verified the MindBand's Bluetooth connection to the recording computer. Two of the subjects whose EDA signals were undetectable also exhibited problems with the MindBand, thus we were not able to record their MindBand either. We then calibrated the subjects' eye gaze using Tobii Studio's 9-point calibration program. We recalibrated any points that showed too much error. Finally, we began the experiment.

⑦ The subjects were asked to watch the first fish tank video and ⑧ start their first task. ⑨ After each task, the subjects were given a paper-based NASA TLX survey instrument [Hart and Staveland, 1988] to fill out that asked them to first rate the task from $1 - 20$ along six dimensions: mental demand, physical demand, temporal demand, performance, effort, and frustration, and then compare each dimension with one another to determine the rank order of their importance. ⑩ Afterward, they watched the next two-minute fish tank video and continued to the next task. ⑪ Once the subjects finished their last task and NASA TLX survey, we removed all of the psychophysiological sensors, and turned off the recordings. ⑫ Finally, we had them fill out a post-test questionnaire where they ranked the tasks they did according to the their own hind-sight perception of

the tasks' difficulty. Subjects were able to go back and refamiliarize themselves with the task codes before ranking them.

2.3.5 Experimental Conditions

Every subject was expected to complete all ten tasks; first the two practice problems and then the eight experimental tasks. To combat any kind of learning effects caused by experience with the tasks, we counter-balanced the task order so that each subject took them in a different order. On average, it took the subjects 1:49 minutes (SD 1:37 minutes) to complete a task. The fastest subject completed one task of the second kind in 9 seconds. The slowest subject completed one task in 8:29 minutes, also of the second kind. Overall, each subject took about 1.5 hours to complete the entire experiment.

Some subjects failed to complete all the tasks before they had to leave. Two missed the final task, and one missed the last two tasks. Fortunately, we had no measurement difficulties with these three subjects.

2.4 Data Analysis

We collected psychophysiological measurements for a total of 116 tasks. We present an overview of each sensor's measurements along with their related cognitive effects in Table 2.1. A detailed list of every measurement we used from the sensors is in Table 2.2.

For each subject's tasks, we also collected the completion time, the NASA TLX score, whether their answer was correct, and the difficulty rank they gave that task at the end of the study. We used the video recordings and the think-aloud protocols to fix any inadvertent mistakes we made during data analysis, which we describe next.

2.4.1 Data Cleaning and Transformation

Biometric data is notoriously noisy and contains large amounts of invalid data that must be cleaned before it can be analyzed.

Table 2.1: Overview of psychophysiological measurements and the effects related to them in literature.

Measure	Previously found effect
Eyetracking	
Pupil size	Cognitive load [Haapalainen et al., 2010], [Klingner, 2010] Memory load [Beatty, 1982]; Mental workload [Iqbal et al., 2004]
Saccades	Mental workload while air traffic control tasks [Brookings et al., 1996]; Evaluation of user interfaces [Goldberg and Kotval, 1999]
Fixations	Cognitive load while solving arithmetical tasks [Ikehara and Crosby, 2005]; Performance during a code review [Uwano et al., 2006, Sharif et al., 2012]; Effort to identify variable identifiers [Sharif et al., 2012]
EEG	
Eye blinks	Visual attention [De Jong and Merckelbach, 1990]; Stress and anxiety level [Doehring, 1957]; Classification of visual demanding tasks during flight [Wilson, 2002]; Mental workload while air traffic control tasks [Brookings et al., 1996]; Mental workload during arithmetic and visual tracking tasks [Ryu and Myung, 2005]
Frequency bands (Alpha, Beta, Gamma, Delta, Theta)	Mental workload during air traffic control tasks [Brookings et al., 1996]; Mental workload during arithmetic and visual tracking task [Ryu and Myung, 2005]; Cognitive task classification [Lee and Tan, 2006]; Auditory awareness [Makeig and Jung, 1996]
Ratios of frequency bands	Memory load during cognitive task [Grimes et al., 2008]; Task engagement index [Kramer, 1991], [Lawrence J. Prinzel et al., 2001], [Berka et al., 2007]; Car driver status in various conditions [Brookhuis and De Waard, 1993]
Attention and Meditation	Cognitive load [Haapalainen et al., 2010]
EDA	
Tonic	Anger and fear [Ax, 1953]; Mood states of bipolar patients [Greco et al., 2012]; Mental workload [Wilson, 2002]; Arousal and engagement [McDuff et al., 2012]
Phasic	Anger and fear [Ax, 1953]; Distinguish stress from cognitive load [Setz et al., 2010]; Arousal and engagement [McDuff et al., 2012]

Eye-Tracker. First, for each data point produced by the eye-tracker, an indication of the validity of the pupil size measurement enabled us to remove the invalid ones. Second, we noticed that the first pupil size measurement of each fixation occurring after a blink was suspiciously larger than the subsequent one (measured just 3.3 ms later). We learned that when your eyes close, even for a

Table 2.2: Psychophysiological measurements used from each of three sensors (abbreviated) (Δ represents the difference to the baseline).

Eyetracking (18)
NumSaccades/Min; SumSaccadeDuration/Min; {Mean, Median, Stdev}SaccadeDuration; NumFixations/Min; SumFixationDuration/Min; {Mean, Median, Stdev}FixationDuration; ΔNumPupilSizeJumps>{0.1mm, 0.2mm, 0.4mm}; MinPupilSize; MaxPupilSize; Δ{Mean, Median, Stdev}PupilSize
EEG (31)
{Min, Max}Attention; {Min, Max}Meditation; Δ{Mean, Stdev}Attention; Δ{Mean, Stdev}Meditation; ΔEyeblinks/Min; $\Delta(\alpha/\beta)$; $\Delta(\alpha/\gamma)$; $\Delta(\alpha/\delta)$; $\Delta(\alpha/\theta)$; $\Delta(\beta/\alpha)$; $\Delta(\beta/\gamma)$; $\Delta(\beta/\delta)$; $\Delta(\beta/\theta)$; $\Delta(\gamma/\alpha)$; $\Delta(\gamma/\beta)$; $\Delta(\gamma/\delta)$; $\Delta(\gamma/\theta)$; $\Delta(\delta/\alpha)$; $\Delta(\delta/\beta)$; $\Delta(\delta/\gamma)$; $\Delta(\delta/\theta)$; $\Delta(\theta/\alpha)$; $\Delta(\theta/\beta)$; $\Delta(\theta/\gamma)$; $\Delta(\theta/\delta)$; $\Delta(\theta/(\alpha+\beta))$; $\Delta(\beta/(\alpha+\theta))$
EDA (7)
ΔMeanPhasicPeakAmpl; ΔNumPhasicPeaks/Min; ΔSumPhasicPeakAmpl/Min; ΔMeanSCL; ΔAUCPhasic; {Min, Max}PeakAmpl

short time, the darkness causes your pupils to open just a little bit. To eliminate this artifact, we ended up eliding each of the first pupil size measurements after a blink.

Next, we compared the distribution of pupil sizes between subjects. We discovered that while each subject's pupil size distribution was Gaussian, the range of pupil sizes was very different. Consequently, to make subjects easier to compare, we standardized the pupil size measurement within participants by subtracting the mean from each value and dividing the difference by the standard deviation.

Pupil size tends to increase up to 0.5 mm under cognitive load, especially when reading difficult material. To find these events, we use a Matlab-based peak finding algorithm to count the number of peaks in the pupil size signal where the pupil size increased at least 0.1, 0.2, and 0.4 mm above its baseline. The baseline is calculated from the minimum and maximum pupil sizes gathered during each task as well as the prior one minute during the fish tank video.

People's eyes move in small jerky movements called saccades, which each take under 75 ms. Someone can only read text when their eye fixates on a location between saccades. We extract the number and duration of a subject's eye saccades and fixations to gain insight into how their eye motion is impacted

when reading material with various cognitive demands. Since every subject works at their own pace, we normalize many of our measurements by time to make them comparable between subjects.

EDA. EDA signals consist of two parts: a low frequency *tonic* signal which changes over a period of minutes, and a higher-frequency *phasic* signal, which takes 1–2 seconds to rise and 2–6 seconds to fall. The tonic component of the EDA signal, or skin conductance level (SCL), is commonly used as a measure of arousal. The phasic component reflects reactions based on external stimuli [Schmidt and Walach, 2000].

To clean our EDA signal, we first subtracted the signal's DC component to base it at 0 µS. We found a lot of noise in the signal from 2 Hz to 4 Hz, so we applied an exponential smoothing filter ($\alpha x(t) + (1-\alpha)x(t-1), \alpha = 0.08$). Next, we used a 5th order, low-pass Butterworth filter set to 0.05 Hz to extract the tonic signal. Since the maximum frequency of a phasic response is 0.33 Hz (the inverse of 6 seconds), we must extract the phasic signal at 0.66 Hz (the Nyquist sampling rate is twice the maximum frequency) to ensure we see the entire phasic response. Fortunately, the exponential smoothing we applied already eliminated the signal above 0.66 Hz, so we were able to use a high-pass version of the same Butterworth filter to extract the phasic signal.

The tonic SCL value must be measured relative to a recent baseline value. We calculate it by subtracting the mean SCL of the EDA signal while the subject watched the fish tank video from the one measured while the subject did each task. The literature distinguishes between spontaneous changes in the EDA signal — called non-specific skin conductance responses (NS-SCRs) — and changes that occur after a specific stimuli — called event-related skin conductance responses (ER-SCRs) [Andreassi, 2007]. These changes are visible as peaks in the phasic signal which we found with a Matlab-based peak finder set to identify peaks with a minimum amplitude of 2 nS [Fowles et al., 1981]. While NS-SCRs occur all the time, the only external stimuli the subjects could have experienced must have come from what they read during their program comprehension tasks. Thus, we can compute the likely number of ER-SCRs by subtracting the number of peaks experienced during the experimental task from the preceding one minute

time period while they watched the fish tank video. We also use the peak finder to extract additional features from the signal, including the peak amplitude, frequency, and area under the curve (AUC) [Ax, 1953], [Setz et al., 2010], and normalize these by time.

EEG. The EEG sensor produces a raw signal sampled at 512 Hz. We first use a Matlab-based 60 Hz notch filter to remove signal noise caused by the overhead lights. To identify various mental states [Berger, 1929], we use Matlab's pwelch function to compute the power spectrum distribution for each of the five familiar brain wave frequency bands: Alpha (α) (8–12 Hz), Beta (β) (12–30 Hz), Gamma (γ) (30–80 Hz), Delta (δ) (0–4 Hz) and Theta (θ) (4–8 Hz) [Handy, 2005]. Since every person has a unique power spectrum distribution, we compute the ratio of each band with one another in order to compare the values between individuals. In addition, inspired by Kramer and Lee [Kramer, 1991], [Lee and Tan, 2006], we compute $Beta(\beta)/(Alpha(\alpha) + Theta(\theta))$ and $Theta(\theta)/(Alpha(\alpha) + Beta(\beta))$ as additional measures of task difficulty.

We found an additional use for the EEG sensor. Due to its placement on the forehead, the sensor is exquisitely sensitive to the motor signals of the face, such as brow furrowing, eyebrow motion, and blinking. Each of these motor activities produces a high amplitude, low frequency signal which is easy to distinguish from neuronal activity. Brookings *et al.* showed that a person's blink rate decreases significantly when tasks become more difficult [Brookings et al., 1996]. Taking advantage of a technique illustrated by Manoilov [Manoilov, 2007], we use a band-pass Butterworth filter to filter our EEG signal from 0.5 Hz to 3 Hz and apply our Matlab-based peak finding algorithm to find peaks that are over 100x stronger than the waveform's average amplitude. These peaks correspond to eye blinks. We calculate the number of blinks per minute and then subtract out the baseline number of blinks during the subject's prior viewing of the fish tank video.

Finally, we extract the pre-computed 1 Hz Attention and Meditation signals from the NeuroSky EEG sensor, and compute the mean, median, standard deviation, minimum, and maximum values for our analysis.

2.4.2 Outcome Measures

We used two outcome measures in our tasks: the NASA Task Load Index (TLX) [Hart and Staveland, 1988] filled out on paper after each task, and a subjective ranking of tasks based on the subject's *post hoc* perception of their difficulty at the end of the experimental session. The NASA TLX is a commonly-used subjective measure for assessing cognitive load [Haapalainen et al., 2010]. After each task, the subject rates its difficulty on six 20-point scales: performance (good/poor), mental demand (low/high), physical demand (low/high), temporal demand (low/high), effort (low/high), and frustration (low/high). Each scale is defined for the subject using Hart and Staveland's instructions [NASA - Ames Research Center, Aerospace Human Factors Research Division, 1986] along with a discussion of their meaning with the experiment administrator. After marking the six ratings, the subject then considers every possible pair of scale names, and is instructed to circle the scale name in each pair which is more important to his experience of workload than the other. We compute the overall NASA TLX score to be the sum of the products of each rating and the tally (0–5) of the number of times it was chosen as more important, and then divided by 15. Three of the authors computed these scores at different times during analysis to ensure we transcribed and calculated them properly.

While the NASA TLX score gives us insight into the subject's mental workload for each question, we were interested in a measure of the subject's summative assessment of task difficulty. To this end, we asked each subject to rank the ten tasks he did from easy to hard (ties were acceptable). A few subjects wrote down additional comments to clarify how they thought about the difficulty (*e.g.*, "The Rectangle tasks were difficult because I am terrible at doing spatial relations in my head.").

To make prediction simpler for our machine learning algorithm, we nominalized the task difficulty ranking as easy or difficult. Low ranks were changed to easy and high ranks were labeled difficult. For scores in the middle, we looked at each subject's additional comments and found that in all but two cases out of 116, subjects clearly expressed where there was an easy/difficult gap in their perception of the tasks' difficulty. For the other two cases, we were able to use

the NASA TLX score to disambiguate (in favor of correlation) because there the NASA TLX score was clearly unambiguous. After nominalizing the task difficulty ranking, our dataset consisted of 51 difficult and 65 easy tasks.

To validate the task difficulty ranking, we confirmed that there was a high correlation between the task difficulty ranking and the NASA TLX scores. A Spearman correlation shows that the NASA TLX score is correlated with the subjects' task difficulty rankings ($r[116] = 0.587$, $p < 0.01$). The NASA TLX easy/difficult boolean was also similar to the Task Difficulty easy/difficult boolean ($\chi^2(1, 116) = 57.954$, $p < 0.01$) with an accuracy of 85%.

As a final step in validating the task difficulty ranking, we looked at the correlation between the ranking and task completion time, since time on task is also a proxy for difficulty [Haapalainen et al., 2010]. A Spearman correlation of $r[116] = 0.724$ ($p < 0.01$) supports this correlation, and thus our choice of task difficulty ranking for our outcome measure.

2.4.3 Machine Learning

Machine learning has been shown to be a promising approach to find links between low-level data capture and high-level phenomena of interest [Bednarik et al., 2012]. We used Weka [Hall et al., 2009], a popular, Java-based machine learning classification toolkit, to develop a set of classifiers that can connect our psychophysiological measures with task difficulty.

There were a number of parameters that could affect the design of the classifier we wished to develop. First, and foremost, was a choice between three types of predictions: by participant, by task, and by participant-task pair. The by-participant classifier would be the most useful in practice — trained on a small set of people doing program comprehension tasks, it could be applied to any new person doing new tasks and still accurately assess task difficulty. Next, in utility, is the by-task classifier; when trained on people doing a set of tasks, it would work well when applied to one of those people doing any new task. Finally, the by-participant-task pair classifier shows the proof-of-concept — trained on a set of people doing programming tasks, it can predict the difficulty of the task as

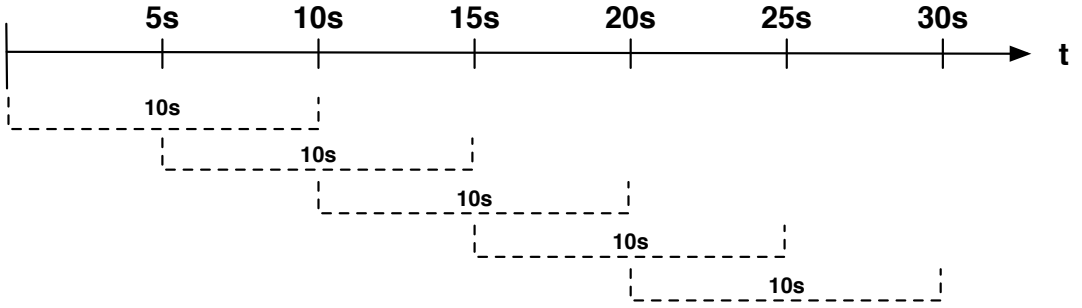


Figure 2.1: Sliding windows of size 10 seconds with 5 second offsets.

perceived by one of those people doing a task that the rest already did. These three predictions were used to stratify the datasets into test and training sets.

Second, was the choice of classification algorithm. We considered Naive Bayes, a J48 decision tree (using Weka's implementation of C4.5 [Quinlan, 1993]), and a Support Vector Model. For our goal of an instantaneous classifier, Naive Bayes was the best choice because of the ease in which its training can be updated on-the-fly, improving its performance as it adjusts to its user.

Third, we can train the classifier on the entire set of data from each participant and task, or divide up the data collection into sliding time windows. This would enable us to create a classifier usable before a developer finished his task that would adjust to his changing psychophysiological conditions. We divided up our data using sliding time windows of sizes from 5 seconds to 60 seconds, sliding 5 seconds between intervals. A demonstration of this is shown in Figure 2.1.

Finally, the last parameter to the machine learning algorithm is to identify the best set of measurements (features) that will be used to train the model [Eivazi and Bednarik, 2011], [Grimes et al., 2008]. We chose to experiment with measurements extracted from every combination of our three sensors (7 possible sets of features). To ensure correct performance for Naive Bayes, we removed five measurements that correlated almost perfectly with measurements we left in.

2.5 Results

This section reports the results of our use of machine learning to define classifiers to predict task difficulty.

2.5.1 Task Difficulty Classification

To evaluate whether we can use psychophysiological measures to predict if a task is easy or difficult (RQ1), we perform a post-hoc analysis that applies machine learning to the data gathered over the whole task period. We used a leave-one-out strategy to create an exhaustive set of test and training folds to train classifiers using all of the sensors for each stratification (by participant, by task, by participant-task). The average precision, recall, and f-measure for the three classifiers we trained is shown on the last row of each section in Table 2.3. The best overall performance comes when predicting a new task with 84.38% precision and 69.79% recall.

2.5.2 Evaluating Sensors

Next, we wished to find out how well each of the three sensors, eye-tracking, EDA, and EEG, could be used to predict task difficulty (RQ2). We trained classifiers on each combination of sensors creating training and test sets for all three predictions (by participant, by task, by participant-task) over the entire dataset. The results are shown in Table 2.3. Considering each sensor by itself, the eye-tracker has the best predictive power for new participants (65.10% f-measure). When predicting a new task, EEG has the highest precision (81.97%), but the eye-tracker has the best recall (66.67%). When predicting a participant-task pair, the eye-tracker comes out on top (66.67% f-measure). Combinations of sensors performed better when predicting new tasks (all sensors) and participant-task pairs (eye-tracker+EDA).

We investigated whether eliminating features from each sensor could help improve the accuracy of our classifiers. We used Weka's CfsSubsetEval algorithm [Quinlan, 1993] to analyze the features for each sensor combination and

Table 2.3: Performance characteristics of classifiers trained on the entire dataset over data from all possible combinations of three sensors to predict a participant, a task, and a participant-task pair. The best measurements for a prediction are bold.

Prediction	Sensors	Precision	Recall	F-Measure
By Participant	**Eye**	**69.16%**	**65.83%**	**65.10%**
	EDA	55.18%	55.77%	51.99%
	EEG	53.05%	56.73%	50.82%
	Eye+EDA	68.37%	64.42%	61.92%
	Eye+EEG	68.58%	63.46%	60.89%
	EDA+EEG	68.02%	64.58%	62.01%
	Eye+EDA+EEG	64.99%	64.58%	62.21%
By Task	Eye	79.17%	66.67%	69.65%
	EDA	75.12%	58.65%	63.80%
	EEG	81.97%	59.62%	63.40%
	Eye+EDA	78.59%	66.35%	70.37%
	Eye+EEG	82.42%	66.35%	69.89%
	EDA+EEG	82.79%	65.63%	69.76%
	Eye+EDA+EEG	**84.38%**	**69.79%**	**73.33%**
By Participant-Task	Eye	66.67%	66.67%	66.67%
	EDA	59.62%	59.62%	59.62%
	EEG	56.73%	56.73%	56.73%
	Eye+EDA	**68.27%**	**68.27%**	**68.27%**
	Eye+EEG	62.50%	62.50%	62.50%
	EDA+EEG	62.50%	62.50%	62.50%
	Eye+EDA+EEG	67.71%	67.71%	67.71%

keep those that correlated highly with the outcome variable and poorly with other features. In some cases, this yielded better performance on our data (*e.g.*, EDA+EEG f-measure rose from 62.01% to 69.73%), however running ANOVA tests on the Weka output failed to show any significant differences between the original and shrunken sets of features. Thus, the improvement we saw may be an artifact of our dataset; capturing additional input data would help establish whether feature elimination will truly improve performance.

2.5.3 Evaluating Time Windows

Finally, to see if we could build a classifier that would be accurate if receiving streaming data from the sensors as the developer worked, we built a set of classifiers trained on sliding time windows (RQ3). However, we needed to find out which time window sizes would work the best. In some cases, the window size was longer than the task data, so we just included the time windows that were available. Figure 2.2 presents the precision for each classifier trained on a particular time window size using all of the available sensors. There appears to be no major differences in the performance of that classifier over the various time windows.

However, we calculated the effects of combining a subset of sensors along with the use of sliding time windows. We found the best classifier for predicting new participants to use just the eye-tracker and the EDA sensor with a time window size of 60 seconds. This performed at 70.46% precision and 62.20% recall, which is just a tiny bit better than using all three sensors or just the eye-tracker. For predicting tasks, the best classifier used just the EDA sensor with a 20 second time window and got a precision of 83.74% and a recall of 64.12%. This performs better than using all three sensors on sliding time window data, but not better than when trained on the entire dataset. When predicting a participant-task, the best classifier used the EDA and EEG sensors with a time window of 55 seconds. This achieved a precision of 100.00% and a recall of 66.13%, which is better than both using all of the sensors and the entire dataset.

2.6 Discussion

The results of our machine learning experiments answer RQ1, demonstrating that it is possible to very accurately predict whether a task is easy or difficult using psychophysiological measures. Using all of the task data, a classifier trained on the three sensors achieves 64.99% precision and 64.58% recall when predicting our nominalized task difficulty measure for new participants. The performance increases to 84.38% precision and 69.79% recall when predicting new tasks, likely

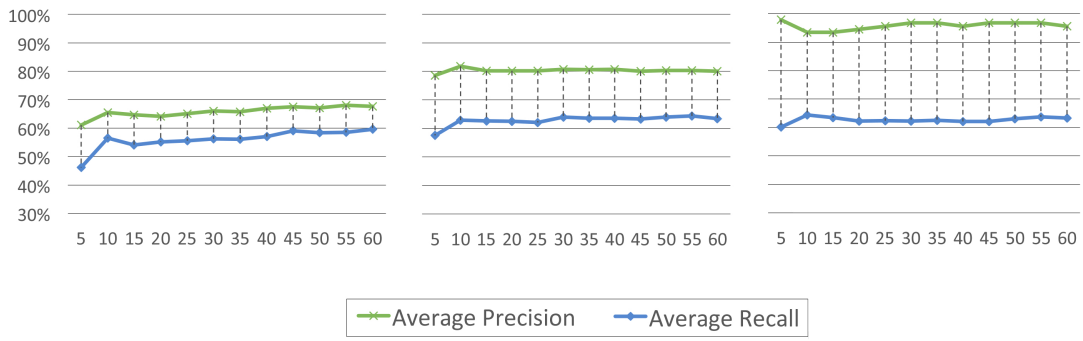


Figure 2.2: Precision and recall using all sensors over time windows of 5–60 seconds.

because the classifier has had a chance to see the participant in action on other tasks. For predicting participant-task pairs, the precision and recall both settle at 67.71%.

Answering RQ2, when we checked which combinations of sensors had the best predictive power, we found that for predicting a new participant, the eye tracker did the best (69.16% precision and 65.83% recall); when predicting a new task, the combination of all three sensors was best (84.38% precision and 69.79% recall); and if predicting a participant-task pair, the pair of eye tracking and EDA sensors was best (68.27% precision and 68.27% recall). Thus, for predicting new participants and new participant-task pairs, it may be better to use a subset of the sensors available to achieve better performance (and save money!).

When we measured the predictive power of classifiers that use sliding time windows, we found that for predicting new participants, the best classifier using all three sensors uses a time window of 55 seconds, and reaches 68.04% precision and 58.55% recall. The best classifier for predicting a new task uses a 30 second time window and reaches a precision of 80.68% and a recall of 64.01%. Finally, for predicting a participant-task pair, the best classifier uses a time window of 55 seconds, and achieves a precision of 96.74% and a recall of 63.73%. Compared with using the entire dataset, dividing the data into sliding time windows is beneficial for predicting new participants and participant-tasks, but not for predicting new tasks.

When we combined the use of sliding time windows with subsets of sensors, we found it possible to improve the precision and recall slightly, compared with using the entire dataset and all of the sensors. The big improvement came for predicting a participant-task pair with just the EDA and EEG sensors (our two lowest cost sensors) and a time window of 55 seconds.

Our work provides an existence proof that answers RQ3. It is possible to use low-cost, off-the-shelf psychophysiological sensors to develop accurate classifiers. The existence of such an indicator should provide many opportunities for new software engineering tools. For instance, it could be used to detect the places in the code that developers have difficulties with while working, and mark them for review or for future refactoring. As pointed out by Bailey and Iqbal, it could also help prevent interruptions during particularly difficult tasks which might require a longer task resumption time [Bailey and Iqbal, 2008].

2.7 Threats to Validity

We describe several threats to the validity of our study in this section.

External Validity. While we feel these results should be generalizable to other kinds of short, code comprehension tasks, more work remains to be done to validate our classifier against tasks that are longer, contain more code, and involve code creation and maintenance. We mitigated this risk somewhat by carefully constructing the tasks to vary their difficulty and effects on various brain functions according to past empirical results on programmers [Détienne, 2002]. We do not claim that these results are generalizable to students, novice software developers, or other broader populations, but feel that our use of professional developers situates this work within a population that creates most of the software in the world.

Internal Validity. During the study, the participants were required to complete a series of small tasks of varying difficulty. We counter-balanced the task order to combat learning effects, but did not have a large enough population to explore every possible order. Thus, some learning effects may have gone unnoticed. Our study took place in a lab setting, thus, our subjects may have performed

differently than in their own work environments. Since the typical effect of lab studies on subjects is to *increase* their performance, due to their desire to please the experimenter, they may have experienced less task difficulty than normal.

While the tasks in our study were not very long (only several minutes), we believe the subjects' behavioral responses (interpreted with their think-aloud narrative) to be fairly typical of software developers working on their own longer-term tasks. As noted in Section 2.3.3, even short, lab-based, experimental tasks like ours can be designed to provoke cognitive difficulties in many of the different functional brain regions that comprise software development skills. More experiments will be required to establish whether the trends we have seen in our data apply to different programming tasks. While there is great individual variability in the performance of software developers (which we also observed in our experimental subjects), there apparently was not enough in the sensor data to impede accurate classification. A study with a larger and more diverse sample of developers should be able to tell us whether the classifiers will be confounded, or confirm their generalizability. Even if the classifiers fail to generalize due to individual participant variation, our use of the Naive Bayes algorithm will support performance improvement through additional classifier training by the participant while he works. Given the great number of hours that developers spend in front of their computers and the potential utility of such an instantaneous classifier of task difficulty, training classifiers for each individual programmer to improve accuracy should be a palatable tradeoff.

Construct Validity. The goal of this study was to investigate the predictive power of multiple psychophysiological measures for task difficulty. A threat to the study is that there are other factors that might either influence the perceived task difficulty or psychophysiological measurements unrelated to task difficulty itself. These include personality traits, private and professional stress, or even the time of day. We tried to mitigate the risk by providing the same quiet environment for every subject, but we may need to investigate these effects in the future. Second, to make predictions simple for our machine learning approach, we categorized the task difficulty ranking into easy or difficult. This binary classification might have lead to better results than if we had used the original

interval scale. Third, our tasks were constructed to be varying shades of difficult, but this is really subjective. We triangulated this difficulty using not just a retrospective ranking of tasks by difficulty, but also through the commonly used NASA TLX score. These measures correlated together quite well, especially when both were converted to nominal form. Thus, we do believe that our task difficulty construct is quite valid.

2.8 Conclusion

Software developers regularly experience difficulties in their work that waste their time and may cause them to introduce bugs into their software. Previous research focused on identifying bug risk using correlations between defects and various software process metrics. Our research, however, is the first to investigate an automated approach using psychophysiological sensor data to detect both *post hoc* and as the developer works, whether a developer perceives that his program comprehension task is difficult. Our experimental results show that we can train a Naive Bayes classifier on short or long time windows with a variety of sensor data to predict whether a new participant will perceive his tasks to be difficult with a precision of over 70% and a recall over 62%. Our results also demonstrate that it is possible to use fewer sensors and still retain the ability to accurately classify task difficulty. Now that we have shown that these classifiers can be built, researchers can leverage them to develop novel programming support tools, allowing them to potentially intervene in time to stop bugs from entering the code.

3

Using (Bio)Metrics to Predict Code Quality Online

Sebastian C. Müller *and Thomas Fritz*
Published at the 38th International Conference on Software Engineering, 2016
Contribution: Study design, data collection, data analysis, and paper writing

Abstract

Finding and fixing code quality concerns, such as defects or poor understandability of code, decreases software development and evolution costs. A common industrial practice to identify code quality concerns early on are code reviews. While code reviews help to identify problems early on, they also impose costs on development and only take place after a code change is already completed. The goal of our research is to automatically identify code quality concerns while a developer is

making a change to the code. By using biometrics, such as heart rate variability, we aim to determine the difficulty a developer experiences working on a part of the code as well as identify and help to fix code quality concerns before they are even committed to the repository.

In a field study with ten professional developers over a two-week period we investigated the use of biometrics to determine code quality concerns. Our results show that biometrics are indeed able to predict quality concerns of parts of the code while a developer is working on, improving upon a naive classifier by more than 26% and outperforming classifiers based on more traditional metrics. In a second study with five professional developers from a different country and company, we found evidence that some of our findings from our initial study can be replicated. Overall, the results from the presented studies suggest that biometrics have the potential to predict code quality concerns online and thus lower development and evolution costs.

3.1 Introduction

A commonly accepted principle in software evolution is that delaying software quality concerns, such as defects or poor understandability of the code, increases the cost of fixing them [Boehm et al., 1976], [Boehm, 1981], [Lehman, 1980], [McConnell, 2004]. Ward Cunningham even went as far as stating that *"every minute spent on not-quite-right code counts as interest on that debt"* [Cunningham, 1993].

Code reviews are one practice that is widely used today to detect code quality problems early on. Code reviews are generally performed by peers after a developer completes the changes for a task and they help to improve code, *e.g.*, its readability, and to find defects [Bacchelli and Bird, 2013], [Rigby et al., 2008], [Bosu et al., 2015], [Kennedy, 2015]. At the same time, code reviews impose costs in terms of time and effort by peers to perform the review. Several automatic approaches to detect code quality concerns have been proposed, for instance, to detect defects [Nagappan et al., 2006], [Zhang et al., 2007], [Nagappan et al., 2008] or code smells [van Emden and Moonen, 2002], [Lanza and Marinescu, 2006]. These approaches generally have two disadvantages in common: first, they

are predominantly based on metrics, such as code churn or module size, that can only be collected after a code change is completed and often require access to further information, such as the history of the code; second, they do not take the individual differences between developers comprehending code into account, such as the ones that exist between novices and experts [Crosby and Stelovsky, 1990].

The goal of our work is to use biometric sensing to overcome these disadvantages and lower the development cost by identifying code quality concerns online — while a developer is still working on the code. Previous research, including one of our earlier studies, has already shown that certain biometric measures, such as heart rate variability (HRV) or electro-dermal activity (EDA), can be linked to task difficulty or difficulty in comprehending small code snippets [Fritz et al., 2014a], [Nakagawa et al., 2014], [Veltman and Gaillard, 1998], [Walter and Porges, 1976]. The general concepts behind these studies are that biometric measures can be used to determine cognitive load — the amount of mental effort required to perform a task — and that the more difficult a task, the higher the cognitive load, and the higher the error rate [Sweller, 1988], [Ayres, 2001]. In our work, we build on top of these concepts and aim to examine the use of biometrics to determine the places in the code that professional developers perceive to be difficult and are therefore more likely to contain quality concerns (errors). This would allow us to automatically perform preemptive code reviews helping developers to commit code with less quality concerns.

To investigate the use of biometrics to predict code quality concerns online, we performed a field study with ten professional developers in a Canadian company over a period of two weeks. We collected a variety of metrics, including biometrics, such as heart rate variability, as well as more traditional metrics, such as code complexity and churn. After each committed change and periodically throughout the study, we asked developers to assess the perceived difficulty of the code elements — methods and classes — they were just working with. Additionally, we collected quality concerns identified in peer code reviews of the committed changes. Amongst other results, our study shows that biometrics outperform more traditional metrics and a naive classifier in predicting a developer's perceived difficulty of code elements while working on these elements. Our analysis also

shows that code elements that are perceived more difficult by developers also end up having more quality concerns found in peer code reviews, which supports our initial assumption. In addition, the results show that biometrics helped to automatically detect 50% of the bugs found in code reviews and outperformed traditional metrics in predicting all quality concerns found in code reviews.

To assess our approach's generalizability, we conducted a second study with five developers in a Swiss company over a period of a week. The results of this study provide evidence that some (but not all) of our findings can be replicated.

In summary, this paper makes the following contributions:

- It presents results of a two-week study with ten developers investigating the use of biometrics in the field to determine code quality concerns and developers' perceived difficulty.
- It provides a comparison between various metrics, showing that biometrics can outperform more traditional metrics in predicting code quality concerns online.
- It presents results of a one-week replication study with five developers from a different company and country.

Overall, the results of our studies suggest that developers' biometrics have potential to identify difficult places in the code and in turn quality concerns and thus might be used to lower the overall software maintenance cost.

3.2 Related Work

Work related to our research can roughly be categorized into three major areas: the manual detection of quality concerns in form of inspections and code reviews, automatic detection based on code, change and interaction metrics, and, more broadly, the use of biometrics in software development.

Manual Detection. The substantial benefits and cost savings of manual software inspection have long been known based on evidence from multiple places [Ackerman et al., 1984], [Ebenau and Strauss, 1994], [Grady and Slack,

1994]. While these results were mostly based on formal inspections, companies today often employ more lightweight and tool supported code review processes that require less time and effort [Techcrunch, 2015]. Several studies have looked into these lighter weight code reviews, in particular their practices, characteristics and outcomes, and shown amongst other results, that these lightweight code reviews still lead to substantial code improvements and the detection of defects [Bacchelli and Bird, 2013], [Rigby et al., 2008], [Bosu et al., 2015]. Overall the results from these studies show that manual code inspection can help to detect many quality concerns soon after code changes were performed and lead to significant cost savings in software evolution. At the same time, manual inspections still require time and effort of peer developers and can only be done after the code was committed or shared for review.

Automatic Detection. There is a myriad of research investigating the automatic detection of code quality concerns. Most of these approaches focus on various software metrics, such as complexity, size or change metrics and their correlation to software defects [Nagappan et al., 2006], [Zhang et al., 2007], [Nagappan and Ball, 2005]. Instead of code and change metrics, researchers have also looked into organizational information to predict defects, for instance the number of developers who touched a file [Weyuker et al., 2008], [Nagappan et al., 2008]. Others have focused on the automatic detection of code smells, predominantly by using code metrics in combination with absolute or relative thresholds or rule sets [Marinescu, 2004], [Alikacem and Sahraoui, 2006], [Munro, 2005], [Moha et al., 2010], but also by mining the change history [Palomba et al., 2013]. Furthermore, there are tools, such as FindBugs or PMD that can help to identify potential quality concerns in the code [FindBugs, 2015], [PMD, 2015]. Most of these approaches only allow for a post-hoc classification and do not take into account the individual differences between developers working on the code. A first step into this direction was taken by Lee *et al.* [Lee et al., 2011] who focused on developers' individual interaction patterns and proposed 56 micro interaction metrics for defect prediction. In a case study, the authors compared defect prediction learners based on change metrics and source code metrics with models based on micro interaction metrics, and found that developers' interaction

patterns, such as the ratio between edits and selects, can significantly improve the defect prediction.

Rather than identifying quality concerns, code metrics have also been used to assess the difficulty of various code-related activities, such as program comprehension. For instance, Curtis *et al.* [Curtis et al., 1979], or Feigenspan *et al.* [Feigenspan et al., 2011], investigated how different kinds of code metrics correlate with developers' performance on maintenance tasks, respectively program comprehension. Closer to our research, Carter *et al.* [Carter and Dewan, 2010a] used interaction logs within the IDE to predict when a developer is stuck, experiencing a lot of difficulties and cannot make any progress.

In contrast to these studies, we are investigating the use of biometric measurements to identify quality concerns. Using biometrics would allow for an online detection that takes into account the individual differences between developers.

Biometrics. In psychology, a broad range of biometric measurements has been investigated and correlated with a person's cognitive states and processes. These biometrics can be roughly categorized into skin-, heart- and breathing-related measurements. Commonly used measurements are electro-dermal activity (EDA) and skin temperature for the skin-, heart rate (HR) and heart rate variability (HRV) for the heart- and the respiratory rate (RR) for breathing-related measurements. For all these measurements, researchers have found correlations to mental and cognitive load/effort, as well as to task difficulty as presented in Table 3.1.

In the context of software engineering, these biometric measurements were used to assess developers' mental load and perceived difficulty while working on small code snippets. Parnin [Parnin, 2011] investigated the potential of electromyography (EMG) to measure sub-vocal utterances and found that this might be used to determine programming task difficulty. In a similar direction, Nakagawa *et al.* [Nakagawa et al., 2014] used Near Infrared Spectroscopy (NIRS) to measure developers' cerebral blood flow (CBF) while working on code comprehension tasks with two difficulty levels. Radevski *et al.* [Radevski et al., 2015] proposed an approach that uses electro-encephalography (EEG) to assess developers' productivity in real time. Finally, in a previous study, we used a combination

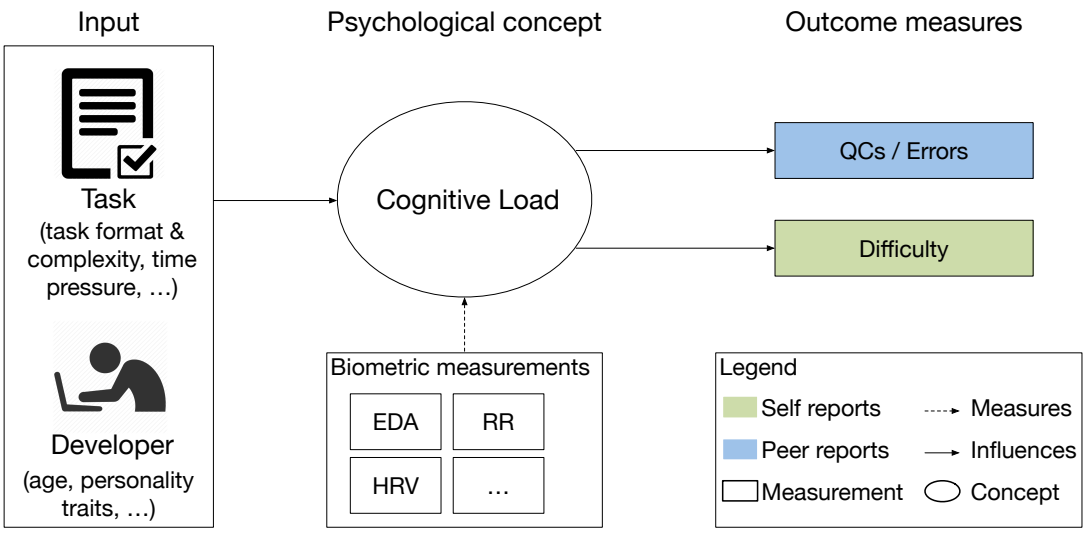


Figure 3.1: Concepts and relationships between input, cognitive load, biometric measurements, and outcome.

of biometric sensors and found that they can be used to predict the difficulty of small code comprehension tasks [Fritz et al., 2014a]. Besides these studies, most research in software engineering using biometric measurements focused on eye tracking technology (*e.g.*, [Bednarik and Tukiainen, 2006], [Crosby and Stelovsky, 1990], [Rodeghero et al., 2014]) or developers' brain activities (*e.g.*, [Ikutani and Uwano, 2014], [Siegmund et al., 2014]) to gain a better understanding of program comprehension. In contrast to all these studies, we focus on the online prediction of code quality concerns and are one of the first ones to perform a longitudinal two-week field study with biometrics sensors.

3.3 Psychological Background

Our work on the use of biometrics builds on top of established psychological research and concepts, including the cognitive load theory. Figure 3.1 illustrates some of these relations relevant to our work. Cognitive load (CL) refers to the required mental effort to perform a task and is composed of intrinsic (*e.g.*, inherent task difficulty), extrinsic (*e.g.*, the way the code is written) and germane

load (effort for processing information) [Sweller et al., 2011]. In general, the more difficult it is to perform a task for an individual, the higher the cognitive load, and in turn, the lower the individual's performance and the higher the error rate [Sweller et al., 2011], [Weast and Neiman, 2010], [Ayres, 2001], [Ko and Myers, 2005]. Previous studies have shown that mental effort and cognitive load can be measured using biometrics (*e.g.*, [Veltman and Gaillard, 1998], [Wilson, 2002]). Based on the links between cognitive load, errors and biometrics, we might be able to use biometric measurements to determine a developer's perceived difficulty when working with a code element and the likelihood of an error being created. Also, since biometrics are linked directly to a developer's cognitive load and thus capture individual differences even for the same task, this approach should be more accurate than proxies that try to capture cognitive load from artifacts.

3.4 Study Method

To investigate the use of biometrics to detect quality concerns online, we analyzed four main research questions:

RQ1 Can we use biometrics to identify places in the code that are perceived to be more difficult by developers?

RQ2 Can we use biometrics to identify code quality concerns found through peer code reviews?

RQ3 How do biometrics compare to more traditional metrics for detecting quality concerns?

RQ4 How sensitive are these biometrics to the individual?

To address our research questions, we conducted a long-term empirical field study with ten professional software developers, working for a medium-sized software development company in Canada. Over the course of two weeks participants worked on their usual tasks in their usual work environment while wearing

biometric sensors. We periodically asked participants to rate the difficulty of the code elements — methods and classes — they were working with on a 6-point Likert scale and collected quality concerns identified in peer code reviews. In addition, we gathered more traditional metrics for comparison purposes[1].

3.4.1 Participants & Sensors

We were able to recruit ten professional software developers from a medium-sized software development company in Canada for our study. The ten participants (nine male, one female) ranged in age from 23 to 45 years and had an average professional software engineering experience of 10.2 years (±6.2), ranging from 3 to 22 years. All study participants worked on the same project, but were split over three different teams that were in charge of different components of the project. All teams followed a similar agile software development process and worked on tasks with similar sizes[2]. Each participant had access to her biometric data and was allowed to quit any time without providing a reason.

We used two biometric sensors for this study: an Empatica E4 wristband [Empatica, 2015] to capture skin- and heart-related measurements, and a SenseCore chest strap[3] to capture skin- (except for EDA), heart-, and breathing-related measurements. Participants were asked to wear the chest strap and optionally also the wristband. We ended up with all ten participants wearing the SenseCore sensor for the two-week study period, and six of them (P01, P04, P05, P06, P07 and P08) also wearing the Empatica wristband in addition.

3.4.2 Study Procedure

At the beginning of the study, we asked each participant to install a small, self-written interaction monitor plugin into their Eclipse IDE that logged each time a developer selected or edited a method or a class within the IDE in combination with the current timestamp. At the same time, the plugin collected a set of

[1] A replication package of this study is available online [Müller and Fritz, 2015a].

[2] For privacy reasons we are not able to disclose more specifics on the company and also substituted code element names throughout the paper.

[3] SenseCore sensors are no longer available due to the company's closure.

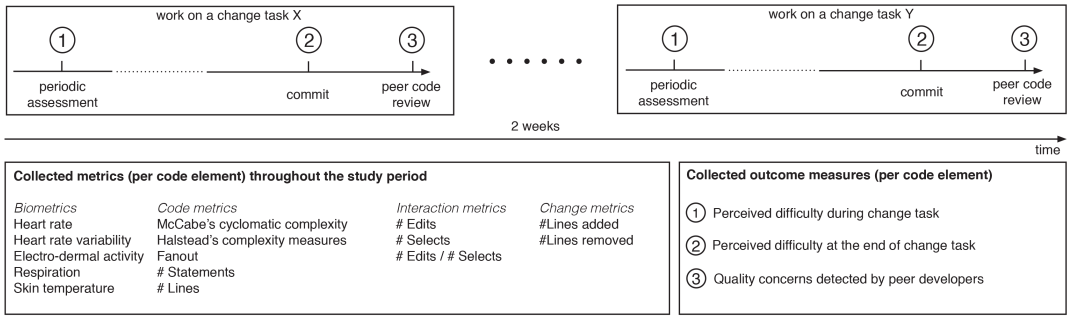


Figure 3.2: Overview of study procedure and collected data.

code metrics for each code element that was selected or edited. After that, we introduced participants to the biometric sensors, gave them the option of either wearing both or just the SenseCore chest strap, and helped them to put the sensor(s) on for the first time. Then we told participants to continue performing their work as usual for the next two weeks while wearing the biometric sensors. An overview of the procedure and data collection for the two-week period is presented in Figure 3.2.

At the end of each workday, we collected the biometric data from each participant and charged the batteries of the sensor(s). Once per day of the study, participants were also asked to watch a two minutes video of fish swimming in a fish tank while wearing the biometric sensor(s). The movie was intended to help participants relax and allow us to record a baseline during the second minute of the two minutes session that we used later on to normalize the captured biometric data. In previous studies [Fritz et al., 2014a], [Müller and Fritz, 2015b] we saw that a person's biometric features drop back to a baseline after about a minute of watching the video.

In addition to the code metrics and interaction logs that we recorded with our Eclipse plugin, we also collected three different types of outcome measures. First, every 90 minutes, the Eclipse plugin prompted participants to answer a small questionnaire within Eclipse that asked them to rate the perceived difficulty of 20 randomly selected code elements that they were working with within the last 90 minutes. Second, every time a participant committed a set of code changes to the repository, we asked the participant to rate the difficulty s/he perceived

Table 3.1: Overview of collected biometrics and their previously found correlations to psychological aspects.

Measurement	Previously found correlations
Heart-related	
Heart rate variability	mental effort [Veltman and Gaillard, 1998]; task difficulty [Walter and Porges, 1976]; mental load [Wilson, 2002, Richter et al., 1998]; task demand [Fairclough et al., 2005]
Heart rate	mental effort [Veltman and Gaillard, 1998]; mental load [Wilson, 2002, Richter et al., 1998]; task difficulty [Walter and Porges, 1976, Anthony et al., 2011, Cornforth et al., 2015]
Breathing-related	
Respiratory rate	mental effort [Veltman and Gaillard, 1998]; task difficulty [Kuznetsov et al., 2011]; task demand [Fairclough et al., 2005]
Skin-related	
Skin temperature	task difficulty [Anthony et al., 2011, Novak et al., 2010, Cornforth et al., 2015]
Electro-dermal activity	mental load [Wilson, 2002, Richter et al., 1998]; task difficulty [Novak et al., 2010, Cornforth et al., 2015]; stress and cognitive load [Setz et al., 2010]

while working on and changing each of the classes and methods in the committed changes. As part of the company's development process, each committed code change was also reviewed by one to three peers. Finally, we collected the outcome of the code reviews for the code changes committed by our participants.

At the end of the study, each participant completed a short demographic background questionnaire.

3.4.3 Metrics and Outcome Measures

We collected four kinds of metrics — biometrics, code metrics, interaction metrics, and change metrics — and three different types of outcome measures.

Biometrics

We used the chest strap and the wristband to collect various biometric measurements that have previously been linked to task difficulty as well as cognitive and mental effort. Table 3.1 provides an overview of the biometrics we captured for this study and the previously found correlations. A complete list of all extracted features can be found in the replication package [Müller and Fritz, 2015a].

To use biometric data for predicting quality concerns of code elements, we had to apply several data segmentation, data cleaning and feature extraction steps. The biometric sensor data is captured as a sequence of data entries with a timestamp and the person's biometric values for that point in time. To map biometric data to code elements, we used the assumption that a developer is thinking about and affected by the code element s/he just selected or edited (see Section 3.8) and therefore segmented the biometric data based on the user interaction log that we captured with our Eclipse plugin. Specifically, we used the time interval from the point in time a developer interacted with a code element C in the IDE up to the point in time s/he interacted with a different element or left the IDE to segment the biometric data and associated the biometric data segment corresponding to this time interval with the code element C. Since a person's heart rate and the phasic part of the EDA signal typically take about one to two seconds to adapt to changes [Berntson et al., 1997], [Sroufe and Waters, 1977], [Schmidt and Walach, 2000], we only considered segments that span at least three seconds in our analysis, *i.e.*, when the developer spent at least three seconds on a code element before moving on, and filtered the biometric data for the first two seconds of the segment to allow for the change in the biometrics to take place.

Heart-related biometrics. For the heart rate, we extracted the mean and the variance of the signal, while for the heart rate variability, we used features that represent the difference in time between two heart beats, such as RMSSD (root mean square of successive differences) or NN50 (the number of pairs of successive beat-to-beat intervals that differ more than 50ms). All these features have been linked to mental effort and load as well as task difficulty [Veltman and Gaillard, 1998], [Anthony et al., 2011].

Breathing-related biometrics. Previous research linked a person's respiratory rate to task difficulty [Kuznetsov et al., 2011], [Fairclough et al., 2005]. We therefore extracted commonly used features, such as the mean respiration rate or the $\log_{10}$ variance of the respiration signal and added them to our feature set.

Skin-related biometrics. For the skin temperature, we extracted features, such as the mean temperature that research has linked to task difficulty [Anthony et al., 2011], [Cornforth et al., 2015]. To extract features from the EDA signal, we used Butterworth filters [Butterworth, 1930] to split it into two parts: the high frequency, fast changing phasic part, and the low frequency, slowly changing tonic part [Schmidt and Walach, 2000]. In a next step, we extracted features related to the peaks in the phasic signal, and features from the tonic part that research has linked to mental load and task difficulty [Wilson, 2002], [Richter et al., 1998], [Novak et al., 2010].

All biometric measurements were normalized using the baseline measurements that we collected during the second minute of the two minutes fish tank movie.

Code, Change and Interaction Metrics

We collected several metrics for code elements — methods and classes — that have previously been associated with difficult code or defects. Most of these metrics were captured with our Eclipse plugin.

Code metrics. For each code element, the plugin calculated code metrics that research has linked to difficulty in program comprehension and code quality. The collected metrics were McCabe's complexity (*e.g.*, [Nagappan et al., 2006]), Halstead's complexity measures (*e.g.*, [Elish and Elish, 2008]), various size metrics (*e.g.*, [Connor, 2011]), and fanout (*e.g.*, [Zimmermann et al., 2007]). Since code metrics might alter when a developer makes changes to a code element, we captured the metrics every time a developer interacted with a code element.

Change metrics. Every time a developer committed a change set to the repository in the study period, we extracted the number of lines added and removed for each code element. Previous research has shown that these metrics can be reliable predictors for defects (*e.g.*, [Moser et al., 2008], [Nagappan and

Ball, 2005]). Due to limited access to the source code repositories in the company, we were only able to collect these metrics on class, and not on method level.

Interaction metrics. Previous research has shown that metrics on interaction data, in particular the ratio between edit and select events, might be used to improve defect prediction and to determine when a developer experiences difficulties (*e.g.*, [Carter and Dewan, 2010a], [Lee et al., 2011]). We therefore collected the number and ratio of edit and select events for each code element.

Outcome Measures

Over the course of the study, we collected three different types of outcome measures.

Perceived difficulty during a change task. Every 90 minutes, participants were prompted with a questionnaire that asked them to rate the difficulty they perceived while working on 20 randomly selected code elements from the previous 90 minutes on a 6-point Likert scale (from 1 = "very easy" to 6 = "very difficult"). For the 20 elements in each questionnaire, we equally balanced the number of methods and classes and the number of edited and selected code elements, unless the participant did not interact with a sufficient number of elements in the previous 90 minutes.

Perceived difficulty at the end of a change task. We manually monitored code repositories. As soon as we noticed that a developer committed a change set to the repository, we asked her/him to rate the difficulty s/he perceived while performing the necessary changes for each class and method that was changed. For this rating, we used the same 6-point Likert scale as for the first outcome measure.

Code quality concerns detected through peer reviews. Each committed change set was typically reviewed by one to three peers shortly after the commit time. The reviewers looked for actual bugs, inadequate documentation or test cases, and violations of coding styles. We collected the results of these code reviews for each change set that was committed by one of the study participants.

We marked a code element as containing a quality concern when at least one was identified in a code review.

3.4.4 Data Collection

Across all study participants and the two weeks of the study, we were able to collect biometric measurements for a total of 116 developer work days (∅=11.6, ±1.8). This resulted in 12.1 GB worth of biometric data, consisting of 40.6 million data points. For all ten study participants, we collected skin temperature, HR, HRV and RR data. For six study participants who volunteered to also wear the Empatica wristband sensor, we were able to collect the EDA as well as a second skin temperature and HR(V) measurement. We decided to take the signal from the SenseCore sensor whenever possible and only rely on the Empatica signal in case the SenseCore signal could not be recorded, since our previous experiences with the two sensors indicate that the SenseCore signal is more accurate.

In addition to the biometric data, we collected perceived difficulty ratings for 1511 methods and 1480 classes. From the 1511 difficulty ratings for methods, 982 were collected while developers were working on a change task, while the rest were collected at commit time. Similarly, 900 classes were rated while working on a change task and 580 at commit time. On average, study participants spent 12.0 minutes on a particular class and 6.8 minutes on a particular method, between two consecutive difficulty ratings that occurred every 90 minutes. Table 3.2 provides an overview of the difficulty ratings we collected for each participant in the study. For all code elements that were changed and committed by one of our participants, we were also able to collect the results of the peer code reviews of these elements. In total, we collected 162 quality concerns, 46 on method level and 116 on class level. We ended up with 95 (16.4%) classes in which a quality concern was found and 485 (83.6%) without any quality concern. Similarly, our dataset consists of 44 (8.3%) methods with a quality concern and 485 (91.7%) methods without any quality concern. Table 3.3 provides an overview of the categories of quality concerns found during code reviews.

Finally, we also collected answers to the demographic questionnaires at the end of the study.

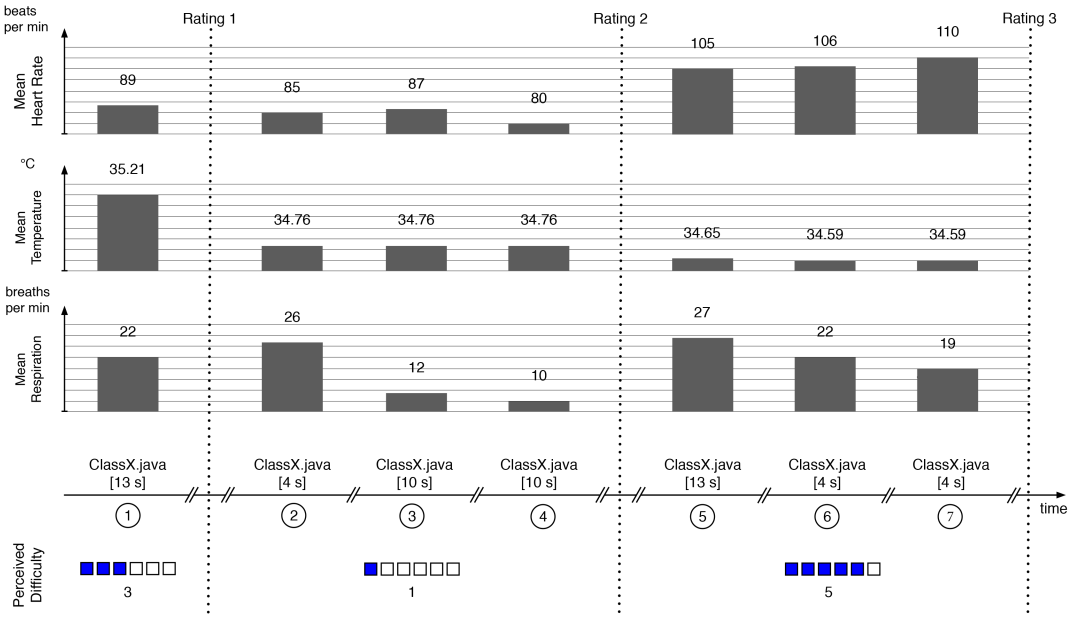


Figure 3.3: Exemplary perceived difficulty rating and biometric data (heart rate, skin temperature and respiration rate) over seven time periods during which participant P02 worked on the class `ClassX.java`.

3.4.5 Data Mapping

Figure 3.3 illustrates some of the ratings and biometric data that we collected for participant P02 on class `ClassX.java` over the course of his/her work on a change task. During the depicted time period, P02 was interacting with `ClassX` seven times. At three points in time during the depicted period, the developer was prompted by our plugin to rate the perceived difficulty while working with this class. For the three ratings, the perceived difficulty changed from three to one to five. While the developer was working on this class, we also captured the biometric measurements as described earlier on. A subset of these metrics is also depicted in Figure 3.3. For each rating by the developer on perceived difficulty during work, we associated the biometric measurements collected between the current and the previous rating. For the example shown in Figure 3.3, we only considered the biometrics captured in interval ① for 'Rating 1', the intervals ② - ④ for 'Rating 2' and intervals ⑤ - ⑦ for 'Rating 3'. In this example, there is

Table 3.2: Number of collected data points for each study participant during and at the end of a change task.

Subject	Methods		Classes		Total
	During	After	During	After	
P01	92	2	77	77	248
P02	106	2	108	12	228
P03	118	71	40	83	312
P04	101	3	72	73	249
P05	137	29	107	15	288
P06	137	74	159	65	435
P07	33	78	34	39	184
P08	90	72	96	51	309
P09	28	69	49	79	225
P10	140	129	158	86	513
Total	982	529	900	580	2991

Table 3.3: Number of quality concerns found in code reviews.

Category	Method	Class	Total
Coding style violation	17	52	69
Bug	14	34	48
Missing test	6	11	17
Insufficient exception handling	5	9	14
Inadequate comments	3	8	11
Other	1	2	3
Total	46	116	162

a visible difference with the mean heart rate being rather low between ‘Rating 1’ and ‘Rating 2’ for which interval the class was perceived easy (1), and the mean heart rate between ‘Rating 2’ and ‘Rating 3’ when the class was perceived more difficult (5). For a developer’s rating of perceived difficulty at the time of a commit, we associated all biometric measurements collected between the current and the previous commit.

For each code metric we captured for a code element and a given time frame — either between two ratings or between two commits — we calculated and collected the metric every time a developer interacted with the element and then

calculated the mean over all interaction instances within the considered time frame.

3.5 Analysis And Results

In the following, we address our research questions by presenting the analysis and results of the gathered data.

3.5.1 Perceived Difficulty and Quality of Code

Figure 3.4 depicts the distribution of collected difficulty ratings. Overall, only very few code elements (3.0%) that developers worked with were perceived as difficult or very difficult, while most (69.3%) were perceived as very easy or easy. To investigate whether and how the perceived difficulty of a code element changes over time, we analyzed a developer's difficulty ratings that we collected for the same code element during or at the end of a change task. While we did not collect multiple ratings for each code element during a change task due to the random selection process, we had 42 (±31.7) cases per developer in which we did. In 51.2% of these cases, the perceived difficulty changed between two consecutive ratings, with 43% of these cases in which the perceived difficulty increased. In most of these cases in which the perceived difficulty changed over the time a developer worked on a change task, code metrics did not change. For instance, the number of lines metric only changed in less than half, and McCabe's cyclomatic complexity only in less than a third of the cases. These results indicate that the perceived difficulty of a code element changes frequently over the course of a change task, and that these changes might often not be reflected in code metrics.

Table 3.4 provides an overview of the number of code elements that participants rated at commit time and the number of quality concerns that were found in these elements in the code reviews. As an example, from the 245 methods that were rated as being very easy, 14 (5.7%) were found to contain a quality concern during code review. The results show that **the more difficult**

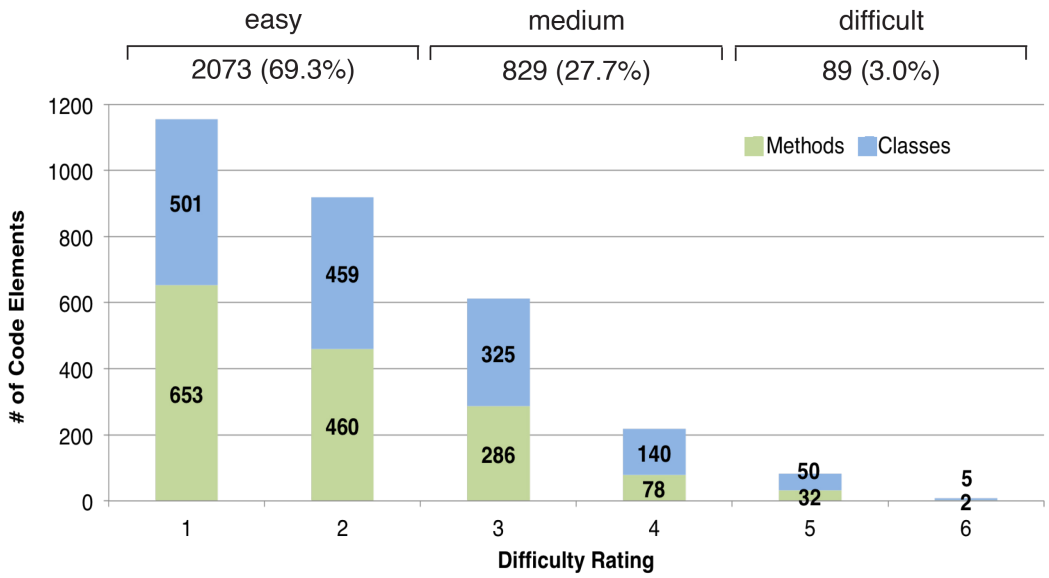


Figure 3.4: Distribution of developers' difficulty ratings of code elements.

a code element is perceived, the higher the likelihood of it containing a quality concern, confirming our initial hypothesis. So while, for example, only 5.7% of the methods perceived as easy had quality concerns, 66.7% (2 out of 3) methods of the difficult elements that were rated as 5 had quality concerns.

3.5.2 Prediction of Code Difficulty and Quality

To answer our research questions, we performed a machine learning experiment. We chose machine learning, since it has been shown to be a good approach for finding links between low-level biometric data and high-level phenomena, such as perceived difficulty and quality concerns [Bednarik et al., 2012].

Machine Learning Approach

We conducted three kinds of predictions on two granularity levels — method and class level — each. In particular, we examined whether we can use machine learning to, first, predict a developer's perceived difficulty of a code element while working on a change task (RQ1), second, predict a developer's perceived

Table 3.4: Quality concerns (QC) found in code elements during code reviews, grouped by perceived difficulty.

	Perceived difficulty					
	1	2	3	4	5	6
Methods						
# reviewed	245	176	90	15	3	0
# with QC	14	17	9	2	2	0
Classes						
# reviewed	174	189	145	47	22	3
# with QC	22	25	29	13	6	0

difficulty of a code element after completing, *i.e.*, committing, the work on a change task (RQ1), and finally, predict whether a code element contains a quality concern found in a code review (RQ2). Since we collected less than four data points for participant P01, P02 and P04 for perceived difficulties of methods after a commit and machine learners need a bigger sample size for reasonable results, we excluded these three participants from this specific prediction analysis.

For the machine learning classifications, we used Weka [Hall et al., 2009], a Java-based machine learning framework. For the classifier, we opted for a Random Forest learner [Breiman, 2001] under the assumption that the non-parametric characteristics of decision trees [Maimon and Rokach, 2006] would fit our collected data, which often exhibited a non-parametric distribution, and because Random Forest learners can deal well with small sample sizes [Qi, 2012]. Studies have shown that for bug prediction based on code and change metrics, the learner should not have a big influence on the performance [Giger et al., 2012], [Lessmann et al., 2008].

We performed a leave-one-out evaluation for each participant separately. This means for each participant and prediction, we trained our classifiers in turn with all data points we captured, except one, and then used the remaining one as test set. We made sure that no identical code elements were in both, the training and the test set. For comparing biometrics with more traditional metrics, we performed each of the six (3 x 2) predictions for five different classifiers: a

classifier based on biometric data, one based on code metrics, one on change metrics, one on interaction metrics, and one that combines all metrics.

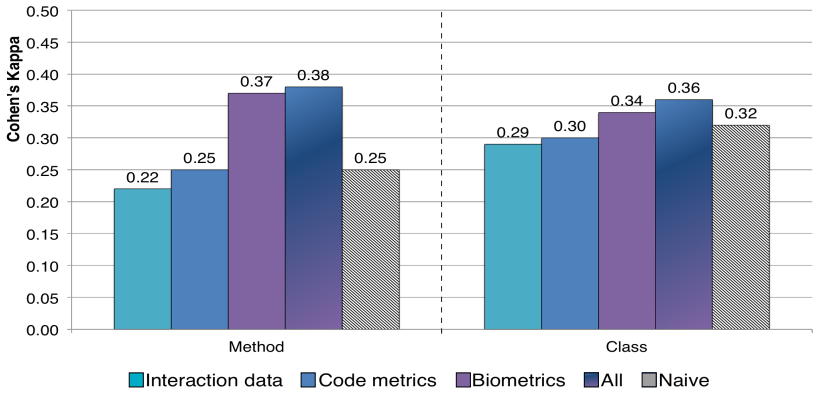
Machine Learning Results

We split our results based on the outcome measure.

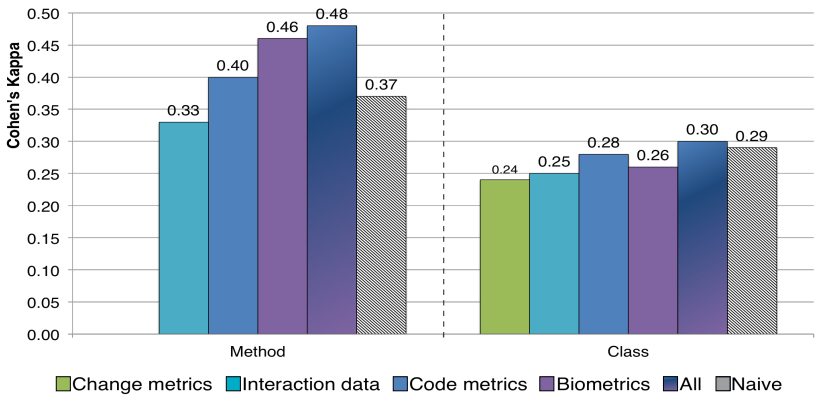
Perceived Difficulty (RQ1, RQ3, RQ4). Figure 3.5 summarizes the results of our machine learning experiments for predicting perceived difficulty. In particular, it presents Cohen's kappa [Cohen, 1960] values for predicting a developer's perceived difficulty of code elements during (Figure 3.5a) and after finishing (Figure 3.5b) a change task. Cohen's kappa measures the agreement between the prediction and the ground truth, taking into account the agreements that might occur by chance. According to Landis and Koch [Landis and Koch, 1977], kappa values from 0 to 0.2 can be considered as slight, from 0.21 to 0.4 as fair, from 0.41 to 0.6 as moderate, from 0.61 to 0.8 as substantial, and from 0.81 to 1 as almost perfect agreement. For comparison reasons, we also added the value for a naive predictor that always predicts the most dominant class but never any other one. To be of practical value, our biometric classifier should be able to outperform this naive predictor.

For three out of the four cases (see Figure 3.5) the biometric classifier outperforms the classifiers that are based on interaction metrics and on code metrics, as well as it outperforms the naive classifier on average by more than 26%. Only a classifier that combines all metrics, including biometrics, achieves better results in these cases, and only for the case of predicting the difficulty of classes after a change task is the biometric classifier worse than a naive one predicting only a dominant class. These results demonstrate the potential that biometrics have in particular for predicting the perceived difficulty of a code element online, while the developer is still working on the change task.

For a more detailed analysis of the results, we chose one case and depicted the confusion matrix for the perceived difficulty prediction on method level in Table 3.5. The matrix shows that in most cases, the predicted difficulty value (1 to 6) is only slightly off of the real value. Finally, Table 3.6 presents the percentage of correct predictions of the biometric classifier for each participant. These



(a) During change task



(b) After change task (at commit)

Figure 3.5: Cohen's Kappa for predicting perceived difficulty for class and method level.

results illustrate the individual differences in the accuracy of predictions. For instance, some participants, such as P10, have a high accuracy for all predictions, while others, such as P07, have a high accuracy for some but not all predictions.

Quality Concerns (RQ2 & RQ3). When predicting whether a code element contains a quality concern (QC) or not (no QC), a biometric classifier performs best over all and even outperforms a classifier that combines all metrics. The top part of Table 3.7 presents the results in terms of precision and recall. We chose

Table 3.5: Confusion matrix for perceived difficulty prediction on method level during the work on a change task. Each cell contains values from each predictor in the order of: all / biometrics / code metrics / interaction metrics / change metrics.

Actual	Prediction					
	1	2	3	4	5	6
1	328/310/284/267/-	59/63/88/87/-	15/27/23/44/-	5/8/9/7/-	0/0/4/3/-	1/0/0/0/-
2	91/83/95/92/-	149/158/127/113/-	43/36/52/64/-	1/7/7/10/-	0/0/3/5/-	0/0/0/0/-
3	49/54/55/56/-	67/58/64/67/-	67/70/57/67/-	8/10/14/6/-	5/4/6/0/-	0/0/0/0/-
4	15/13/21/11/-	13/16/17/19/-	19/19/16/20/-	13/13/7/12/-	3/2/2/1/-	0/0/0/0/-
5	7/4/13/13/-	3/5/3/6/-	7/4/9/6/-	1/2/3/3/-/	11/13/0/1/-	0/1/1/0/-
6	0/1/1/0/-	0/0/0/1/-	1/0/1/1/-	0/0/0/0/-	0/1/0/0/-	1/0/0/0/-

Table 3.6: Percentage of correct predictions per participant using biometrics.

Sub.	**Difficulty (During)**		**Difficulty (After)**		**Quality Concerns**	
	Method	Class	Method	Class	Method	Class
P01	64.1%	71.4%		64.9%		57.1%
P02	59.4%	53.7%		25.0%		83.3%
P03	55.1%	40.0%	53.5%	30.1%	97.2%	84.3%
P04	50.5%	52.8%		41.1%		43.8%
P05	43.1%	39.3%	37.9%	6.7%	96.6%	66.7%
P06	57.7%	49.1%	62.2%	30.8%	60.8%	58.5%
P07	33.3%	20.6%	61.5%	56.4%	66.7%	51.3%
P08	60.0%	37.5%	51.2%	39.2%	88.9%	70.6%
P09	53.6%	44.9%	68.1%	34.2%	53.6%	70.9%
P10	77.1%	81.0%	88.4%	86.1%	65.9%	77.9%
All	57.4%	53.3%	65.7%	46.9%	72.8%	66.0%

precision and recall instead of F-score to highlight the tradeoff between the two. The biometric classifier is able to correctly identify 17 out of 44 (38.6%) code elements with a quality concerns on method level and 38 out of 95 (40%) on class level. At the same time, the precision is not very high with 13.0%, respectively 22.0%, and further research is needed to examine this in more detail.

Table 3.8 provides more details on the percentage of correct classifications for each quality concern category. The data reveals that, with the exception of the "Other" category, the level of correctness is in a similar range for each category. Particularly interesting is the "Bug" category that shows that our biometric classifier is able to identify half of all bugs found in code reviews.

Table 3.7: Results for quality concern (QC) prediction within & across participants in (P: precision, R: recall, ud.: undefined). Bold values accent the best result in each category.

	Metric	Method				Class			
		QC		no QC		QC		no QC	
		P	**R**	**P**	**R**	**P**	**R**	**P**	**R**
Within	All	11.5	27.3	92.3	80.8	17.6	23.2	84.0	78.8
	Biometric	**13.0**	**38.6**	**93.1**	76.2	**22.0**	**40.0**	**86.0**	72.2
	Code	7.9	20.5	91.4	78.0	16.9	29.5	83.8	71.5
	Interaction	0	0	91.1	**94.6**	19.8	16.8	84.2	**86.6**
	Change					17.3	18.9	83.8	82.3
Across	All	**9.7**	**63.6**	**93.1**	45.2	17.5	**30.5**	84.1	71.8
	Biometric	8.3	56.8	91.3	41.8	15.4	22.1	83.3	76.3
	Code	8.1	50.0	91.2	47.9	**20.0**	20.0	**84.3**	84.3
	Interaction	0	0	91.4	**98.1**	16.7	3.2	83.6	96.9
	Change					0	ud.	83.6	**100.0**

Table 3.8: Percentage of correct quality concern predictions by category using a biometric classifier.

Quality Concern Category	**% Correct**	
	Method	**Class**
Coding style	23.5	38.5
Bug	50.0	47.1
Missing tests	50.0	45.5
Insufficient exception handling	20.0	55.6
Inadequate comments	33.3	62.5
Other	100.0	0.0

Within vs. Across Participant (RQ4). To investigate how sensitive the biometrics are to an individual, we performed a second machine learning experiment, in which we trained the classifiers not on each participant individually, but on data from all participants. We again used a leave-one-out approach to train the machine learning classifiers in turn for each participant, except one, and then used the remaining one as test set. We made sure that no code element was in both, the test and training set.

The results for predicting perceived difficulty either during or after a change task are very low. Cohen's kappa values were very close to or well below 0, showing that the predictive power of the classifiers is not any better than chance. For predicting quality concerns with a biometric classifier across participants, the results, however, are better and in some cases even outperform the prediction based on individual classifiers as presented in the bottom half of Table 3.7. The recall values on quality concern predictions on method level are significantly higher than the ones achieved by a within participant classification. However, this comes with a cost and the precision is generally lower and the recall for the code elements that do not contain a quality concern also decreased significantly. We hypothesize that the classifiers trained with data across individuals tend to predict more often that a code element contains a quality concern, since this case is represented in the training set more often, compared to the training set for each participant individually.

3.6 Replication Study

Since there are many factors that might influence the study findings, such as the development process or the source code to name just a few, we performed a second smaller and shorter study. For this study, we collected similar but less data from five professional developers of a medium-sized software development company in Switzerland[4].

Study Method & Participants. For this second study, we were able to recruit five professional software developers, working at a medium-sized software development company in Switzerland. The five study participants worked in four different teams and each team worked on a different product. The study participants were all male, ranged in age from 25 to 30 years (∅=28.0, ±2.3) and had an average professional software development experience of 5.8 years (±2.5).

We followed the study method from our first study to collect metrics and outcome measures. All five participants used the SenseCore chest strap sensor

[4]For privacy reasons we are not able to disclose more specific information about the company.

Table 3.9: Number of collected data points per participant of the second study during and at the end of a change task.

Subject	Methods		Classes		Total
	during	after	during	after	
P11	132	-	165	101	398
P12	30	-	50	28	108
P13	23	-	60	60	143
P14	46	-	37	16	99
P15	7	-	16	9	32
Total	238	-	328	214	780

for approximately one week. Over all participants, we collected data for a total of 25 work days (∅=5.0, ±1.6), including 2.8 GB of biometric data that consists of 11.6 million data points.

Table 3.9 provides an overview of all data points collected during the second study. On average a study participant spend 3.4 minutes per method and 6.8 minutes per class between two consecutive difficulty ratings that occurred every 90 minutes.

Differences & Limitations. The teams in our main study and in the replication study followed a similar development process and developers worked on tasks with similar size. In contrast to the first study, the replication study only lasted one week due to time constraints of the participants. We also only had limited access to the code repositories and thus we were not able to collect perceived difficulty ratings on method level at the end of a change task and we were not able to collect any change metrics. Due to the lack of a code review process in the company, we were also not able to collect any data on quality concerns found in peer reviews. While these differences do not allow us to perform the same analysis, it still allows us to replicate some of the analysis in a vastly different setting. Especially in light of the effort and difficulty to find professional software developers that provide us access to their repositories and that are willing to wear biometric sensors for an extended period of time, we believe this is a reasonable step for an initial replication.

Table 3.10: Cohen's kappa for perceived difficulty prediction for the second study.

Metric	Method (during)	Class (during)	Class (after)
Interaction	0.17	0.11	0.19
Code	0.25	**0.22**	0.27
Biometrics	0.20	0.16	0.37
All	**0.29**	**0.22**	**0.38**
Naive	0.06	0.14	0.07

Machine Learning Predictions & Results. For the data collected during the second study, we extracted the same features from the data and performed the same leave-one-out within participant predictions as we did for the first study. Table 3.10 summaries the results of the predictions. In the second study, the biometric classifier outperforms classifiers based on interaction or code metrics in the 'after commit' case, *i.e.*, after the code changes for a task were finished, with an improvement of more than 37%. In the other two cases for predicting difficulty during a change task, the classifier based on code metrics performs better, but the biometrics classifier is still significantly better than the naive classifier. Similar to the results of the first study, the classifier that incorporates all the different metrics is the best classifier.

In summary, the results of the second study provide initial evidence that we can replicate some of our findings from our main study, but not all. There are many potential reasons for the differences in findings, one of which is that we only collected about half the time of biometric data per code element and rating. Further studies are needed to investigate these aspects in more detail.

3.7 Discussion

In this section we discuss the results of our study as well as their implications on practice and further research.

Predicting Code Quality Online. Our study is the first longer-term study in a real-world software development context with biometric sensors that provides

evidence on the feasibility of using these sensors in the field. The results show that it is possible to predict quality concerns and perceived difficulty of code elements with higher accuracy than traditional metrics in most cases, even despite the noise in professional work environments. Biometrics, different to traditional metrics, allow for online — while the developer is still working on the code — measures, and thus, for example, to prevent bugs from ever being committed by focusing developers' attention on these parts, without requiring access to repositories. Biometrics also factor in developers' individual differences that are not captured by traditional metrics, and thus should provide more accurate results, in particular when they can be trained on each individual.

Our results also show that code elements that are perceived more difficult are also more likely to contain quality concerns. This adds to existing evidence that the difficulty a developer experiences when working on a code element can have a strong influence on the quality concerns the developer creates or adds when changing the code element for the task at hand. Consistently, our results suggest that it is possible to use biometrics not only for predicting perceived difficulty, but also quality concerns identified in peer code reviews.

While the precision for identifying quality concerns in our study could be higher, the fast technology advances leading to more accurate and less noise-sensitive sensors should soon lead to an increase in precision and the value of biometrics in this context. Also, we performed the data analysis for this study retrospectively, but the sensors we used already support real-time data transmission. Since the predictions only require short time windows of a few biometric features, almost instantaneous feedback should soon be possible.

While our smaller scale replication study provides evidence that some of our initial findings can be replicated in other settings, it did not confirm all of our findings. Even though, the biometric classifier still always outperformed the naive classifier, in two out of the three predictions, the biometric classifier was outperformed by a classifier based on code metrics. There are many potential reasons for this, *e.g.*, the development phase, the source code structure, the developers' personalities, or even just the fact that the two studies were performed on different continents. Especially given the sensitivity of biometric sensors,

further, longer term studies are planned to investigate these aspects in more detail.

Tool Support. Our results open up new opportunities for providing tool support. Since we are able to predict early on — while developers are still writing code — whether a code element contains a quality concern, we might be able to help developers and prevent them from ever submitting code with quality concerns to the repository. This could be done by highlighting the affected code element(s) to the developer before s/he commits them and suggesting to spend additional time reviewing. Similarly, one can use this information to suggest which parts of the code might benefit most from a peer code review and prioritize them. Biometric data could also be used to detect when a developer is experiencing difficulties within the code and to provide interactive and immediate feedback ranging from a recommendation to talk to a co-worker to taking a break and continuing later on.

While the study results show that it is possible to detect when a developer experiences difficulties and determine the corresponding code elements to be able to provide the discussed tool support, more research is needed to assess how to best present this information to developers, especially without creating frustration. Also, to provide this kind of tool support the biometric data needs to be collected continuously and transferred in real-time which poses challenges due to sensor invasiveness and more as discussed below.

Challenges. Biometric sensors that have the capability to collect the fine-grained data needed for the kind of study presented here are still under development and pose several challenges due to their usability, invasiveness and the data sensitivity. For our study, we always made sure to have one researcher on site to support participants and we chose sensors that could be worn for several weeks without being too invasive. With the recent advances in sensor technology, the physical invasiveness will decrease even further in the near future. At the same time, more privacy and ethical concerns have to be addressed and investigated, especially since these sensors can be used to collect huge amounts of very sensitive, health-related data. For all these reasons, recruiting study participants who are willing to wear such sensors for weeks and agree to collect a lot of personal data

was also very tedious and time consuming, but in the near future, people might almost automatically collect similar data when wearing watches, such as the Apple Watch [AppleWatch, 2015].

3.8 Threats to Validity

There are several threats to the validity of our study.

External Validity. The generalizability of our findings is limited in many ways, such as the limited number of participants and companies in our study, the focus on Java code and the use of the Eclipse IDE, or the limited number of code elements developers work with and perceive as difficult or very difficult. We tried to mitigate this risk by replicating our initial study and also by collecting data from professional developers in the field, working in different teams and even companies, over a longer period of time and on industrial project code. However, due the broad spectrum of software development differing in aspects, such as the development process, the change task size, the programming languages, the team size, and the development phase the team is in to name just a few, further studies are needed to investigate their implications on the use of biometrics.

Internal Validity. In one part of this study, we used biometrics to predict the quality and difficulty of code elements. A threat to the study is that the data captured with biometric sensors might be affected by other aspects than the perceived difficulty, such as the study participants' personality traits or their general stress level. To mitigate this risk, we used the fish tank videos to capture a baseline each day and normalize the data with it.

Construct Validity. Using the interaction log as an approximation of the code elements a developer might currently be thinking about and using this to segment the biometric data also poses a threat to the validity of our study. However, given the current technologies, this was the best approximation that was also feasible. Eye-tracking devices might provide even more accurate and richer data on which code elements a developer is looking at and thinking about as another study has shown [Kevic et al., 2015], but eye-tracking devices are currently too

expensive or invasive to be used in a long-term field study of this size. Another threat to validity is the use of developers' self-reports, since they might not always accurately represent their experienced difficulty. Finally, our comparison to traditional metrics is limited due to the lack of access to the necessary data and repositories, and future studies are needed to also examine other metrics linked to code quality, such as code churn between multiple versions.

3.9 Conclusion

There is a broad range of tools and research that focuses on the identification of code quality concerns. Yet most of these approaches only allow for a post-hoc assessment and do not take individual differences of developers into account, such as different expertise or experience. In this paper, we presented a first two-week field study on the use of biometric sensors to identify code quality concerns while a developer is working on the code. The results of our study are promising, suggesting that developers' biometrics can indeed be used to determine the perceived difficulty of code elements and furthermore to identify places in the code that end up with code quality concerns, such as bugs. A second smaller replication study we conducted also confirmed some of our findings on the automatic determination of difficult parts in the code. These results open up new opportunities to support developers when they are experiencing difficulties in the code and to fix quality concerns as early as possible, even right when they are being created. With the recent advances in biometric sensing technologies, and their decrease in invasiveness, we might soon be able to collect biometric data on each developer just like we are now already able to collect interaction data. However, this also opens up a discussion on privacy concerns and more research is needed to investigate a feasible solution.

Acknowledgments

The authors would like to thank the study participants. This work was funded in part by ABB and the SNF project "Essentials (People-centric Essentials for Software Evolution)" (project no. 200020_153129).

4

Stuck and Frustrated or In Flow and Happy: Sensing Developers' Emotions and Progress

Sebastian C. Müller *and Thomas Fritz*

Published at the 37th International Conference on Software Engineering, 2015

Contribution: Study design, data collection, data analysis, and paper writing

Abstract

Software developers working on change tasks commonly experience a broad range of emotions, ranging from happiness all the way to frustration and anger. Research, primarily in psychology, has shown that for certain kinds of tasks, emotions correlate with progress and that biometric measures, such as electro-

dermal activity and electroencephalography data, might be used to distinguish between emotions. In our research, we are building on this work and investigate developers' emotions, progress and the use of biometric measures to classify them in the context of software change tasks. We conducted a lab study with 17 participants working on two change tasks each. Participants were wearing three biometric sensors and had to periodically assess their emotions and progress. The results show that the wide range of emotions experienced by developers is correlated with their perceived progress on the change tasks. Our analysis also shows that we can build a classifier to distinguish between positive and negative emotions in 71.36% and between low and high progress in 67.70% of all cases. These results open up opportunities for improving a developer's productivity. For instance, one could use such a classifier for providing recommendations at opportune moments when a developer is stuck and making no progress.

4.1 Introduction

Frustration, anger, happiness and enthusiasm are emotions that software developers frequently experience during their work [Wrobel, 2013]. These emotions are commonly intertwined with the progress one makes, such as experiencing positive emotions leading to more progress [Brief and Weiss, 2002] or the state of being stuck and making no progress leading to frustration [Burleson and Picard, 2004]. Research in psychology has already shown that there is a correlation between these two dimensions, the emotions and the progress people experience for certain kinds of tasks (*e.g.*, [Lawson, 1965]). To help ensure a developer's time is spent as productive as possible, an indicator for a developer's emotions could thus be used to prevent interruptions when a developer is "in flow", making a lot of progress and should not be disturbed, or to provide recommendations at opportune moments when the developer is getting frustrated and close to being stuck.

With the recent advances in biometric (*aka.* psychophysiological) sensor technology, an increasing amount of research in psychology has shown that a person's biometric features, such as skin temperature, facial expression or respi-

ration rate, can be used to detect and distinguish between emotions (*e.g.*, [Picard et al., 2001], [Reuderink et al., 2009]). Psychology research has also shown that biometric measures can be used to determine a flow or stuck state (*e.g.*, [Burleson and Picard, 2004], [Muldner et al., 2010]). However, these studies are focused on small analytical tasks or physics exercises and do not provide any evidence on its applicability to software development tasks, in particular, given the complexity and emotions as well as cognitive skills these kinds of tasks stress in humans.

In software engineering, only little research has focused on developers' emotions and the use of biometric measures. For emotions, researchers have looked at the emotions that developers experience [Shaw, 2004], [Wrobel, 2013], how they might affect productivity [Graziotin et al., 2013], [Khan et al., 2011], and whether one could use interaction logs to predict them [Carter and Dewan, 2010b, Khan et al., 2013]. Using biometric sensors, in particular eye-tracking and fMRI, researchers have mainly studied how software developers comprehend code or use tools [Bednarik and Tukiainen, 2006], [Siegmund et al., 2014], [Sharif and Maletic, 2010a]. In a previous study, we looked at the use of biometric sensors to assess the difficulty of small code comprehension tasks [Fritz et al., 2014a]. In the research presented in this paper, we built upon existing work in software engineering and psychology and further investigate emotions and progress developers experience, as well as the use of biometric sensors to predict them in the context of change tasks. In particular, we are interested in the following three research questions:

RQ1: What is the range of developers' emotions during change tasks and are developers' emotions correlated with their perceived progress?

RQ2: What are aspects and practices that affect developers' emotions and progress during change tasks?

RQ3: Can we use biometric sensors to determine developers' emotions and progress during change tasks?

To address our research questions, we performed a study with 17 participants. In this study, participants worked on two change tasks for 30 minutes each while

we recorded various biometric measures and periodically probed the participants for their emotions and progress. The results of our study show that developers experience a broad range of positive and negative emotions during change tasks that are similar to the ones experienced in other situations and that these emotions are highly correlated with progress, further supporting Graziotin *et al.*'s finding [Graziotin et al., 2013]. The results also show that the localization and understanding of relevant code are the most common aspects for emotions and progress to change. Using the biometric data gathered throughout the study, we trained a machine learning classifier that is able to distinguish between positive and negative emotions with an accuracy of 71.36% and between low and high progress with an accuracy of 67.70%.

This paper makes the following contributions:

- It presents and discusses the results of a study on the emotions and progress software developers have while working on change tasks.
- It presents an approach based on biometric measures to classify a developer's emotions and perceived progress during software development change tasks.

The results of our study suggest that we might be able to use biometric sensors to determine a developer's emotion and progress while working. This opens up a lot of opportunities for improving a developer's productivity.

4.2 Related Work

Work related to our research can be broadly categorized into four areas: general research on emotions and biometrics, research on biometric sensors in software engineering, research on developers' emotions and performance, and research on classifying progress.

4.2.1 Emotions and Biometrics

Research on emotions has a long history in psychology. Many theories and terminologies have been introduced along with several approaches to quantify

emotions [Ekkekakis, 2012]. A widely used approach by Russel [Russell, 1980] differentiates between two cognitive dimensions of emotions: pleasure-displeasure and arousal-sleep. Today, these two dimensions are commonly called valence and arousal [Russell, 2003]. While the valence dimension is considered as the positive or negative character of an emotion [Colombetti, 2005], the arousal dimension indicates the amount of activation and excitement associated with an emotion [Russell, 2003]. In this paper, we generally adapt this terminology and refer to emotions with negative valence as negative emotions and emotions with positive valence as positive emotions.

To measure emotions, a broad range of research in psychology has explored the use of biometric sensors to measure the changes in the body caused by emotions. One of the most common emotions investigated through the use of biometric sensors is frustration. Researchers, for instance, induced frustration by manipulating computer games and measured the effect on the user with biometric sensors. Thereby, they found correlations between frustration and electro-dermal activity (EDA), blood volume pulse (BVP), electroencephalographic (EEG) activity, and muscle tension (*e.g.*, [Riseberg et al., 1998], [Reuderink et al., 2013], [Scheirer et al., 2002]). In other studies, researchers found correlations between self-reported frustration levels and skin conductance or facial EMG while playing games or performing small tasks (*e.g.*, [Kapoor et al., 2007], [Hazlett, 2003]).

To distinguish between various emotions, early research by Ekman *et al.* [Ekman et al., 1983] was able to find differences in biometric signals for four negative emotions. More recently, similar studies were conducted that showed how BVP, EDA, respiration rate, or EEG can be used to distinguish between various emotions, such as anger, fear, sadness, disgust, happiness or surprise (*e.g.*, [Picard et al., 2001], [Chanel et al., 2008], [Murugappan et al., 2008], [Mandryk et al., 2006]).

Instead of distinguishing between different emotions, researchers have also used various biometric sensors to generally distinguish between positive and negative emotions. For instance, Leite *et al.* [Leite et al., 2013] used EDA to measure children's affective state while playing chess, finding that negative affective states

are generally associated with an increased EDA signal that exhibits a lower variation. Reuderink *et al.* [Reuderink et al., 2013] found that EEG measures are correlated with the valence and arousal dimension when they studied subjects playing computer games and induced emotions through the use of non-responsive controllers. Muldner *et al.* [Muldner et al., 2009] found that the pupil size changes with negative and positive affect when they studied subjects solving exercises in physics and varied the affect. Finally, Drachen *et al.* [Drachen et al., 2010] used a combination of biometric measures while participants played a computer game and found that heart rate and EDA are correlated with self-reported negative/positive affect.

In our research, we build upon these previous findings, but focus on software developers performing realistic change tasks that stress a broad range of emotions and cognitive skills. Additionally, we investigate the use of such biometric sensors to predict progress.

4.2.2 Biometrics in Software Engineering

Only few studies in software engineering make use of biometric technology. Most of these focus on the use of eye tracking to examine program comprehension. For instance, Crosby *et al.* [Crosby and Stelovsky, 1990] and Bednarik *et al.* [Bednarik and Tukiainen, 2006] used an eye tracker to study how experienced and less experienced developers understand source code. Similarly, Sharif *et al.* [Sharif and Maletic, 2010a] relied on eye tracking technology to investigate how different identifier naming conventions influence program comprehension by examining the visual effort spent on identifiers.

Very few studies used other biometric sensors. Siegmund *et al.* [Siegmund et al., 2014] used functional magnetic resonance imaging (fMRI) to examine the brain regions that are activated during small program comprehension tasks. Parnin [Parnin, 2011] investigated the use of electromyography to measure subvocal utterances and found that this could be used to determine programming task difficulty. Finally, in a previous study, we used a combination of biometric sensors and found that they can be used to assess the difficulty of small code comprehension tasks [Fritz et al., 2014a].

In contrast to these studies, we focus on the use of biometric sensors to classify developers' emotions and progress during change tasks.

4.2.3 Software Developers' Emotion & Progress

A few studies have investigated the emotions that software developers experience and how these emotions affect their progress and productivity. Early on, Shaw [Shaw, 2004] observed 12 undergraduate students working on a software engineering project and found that the self-reported emotions can change drastically within 48 hours. Similarly, Wrobel *et al.* [Wrobel, 2013] conducted a survey to investigate how emotions impact software developers' effectiveness at work. They found that frustration is the most frequent negative emotion that also disturbs high productivity, and that for some people negative emotions can have a positive effect on productivity. Graziotin *et al.* [Graziotin et al., 2013] conducted an empirical study to investigate whether valence, arousal and dominance correlate with the self-reported performance of software developers. In their study, they observed 8 developers working on a software development task for 90 minutes, asked them every 10 minutes about their emotions and productivity and found that valence and dominance are positively correlated with their productivity. In a second study, Graziotin *et al.* [Graziotin et al., 2014] observed 42 computer science students to find a relationship between affective states and creativity as well as analytic problem-solving skills. The study participants had to perform two tasks, an analytical one and a creative one, and affective states were assessed through a questionnaire. The results imply that developers with positive affect are significantly better problem solvers.

Different to these studies, Kahn *et al.* actively induced moods through videos that developers had to watch or influenced developers' arousal through physical exercises and found that developers' emotions have an influence on debugging performance [Khan et al., 2011]. Closer to our research, Khan *et al.* [Khan et al., 2013] also conducted two studies that focus on measuring mood with keyboard and mouse input. While in the first study, mood was self-reported, the second study induced mood through different kinds of music and an EDA sensor was used to differentiate between high and low arousal. The authors found an

individual correlation between self-reported or induced mood and keyboard and mouse input, but no generic measure.

In our work, we extend results of earlier studies by providing more evidence on the correlation of emotions and progress, the aspects and practices that affect these and, in particular, how biometric measures can be used to classify self-reported emotions and progress during change tasks.

4.2.4 Classifying Progress

Research in psychology has shown that the state of being stuck and making no progress is frequently associated with negative emotions, while a state of flow and making lots of progress is frequently associated with positive emotions [Burleson and Picard, 2004]. Only few approaches try to exploit this relationship and use biometric sensors to determine when people are in a state of being stuck or in flow. Muldner *et al.* [Muldner et al., 2010], for instance, used four different sensors, a posture-chair, a skin conductance sensor, a pressure mouse, and an eye tracker, to determine the so called "*yes*-events" — brief expressions of positive affect — while students were solving physics exercises. They found that students had a larger pupillary response and a higher level of arousal during a yes-event, compared to neutral conditions. Our study is different in that we focus on realistic software change tasks and on emotions and progress.

In the field of software engineering, Carter *et al.* [Carter and Dewan, 2010b] tried to automatically determine moments in which a programmer is stuck based on IDE interaction logs and machine learning. In contrast to this work, we focus on biometric measures that are independent of an IDE and also differentiate between low and high progress as well as negative and positive emotions.

4.3 Study Method and Participants

To learn about developers' emotions and progress when performing change tasks and to address our research questions, we conducted a study with 17 developers. During the study, we had study participants work on two change tasks, while

wearing biometric sensors. Additionally, we periodically asked them to assess their emotions and perceived progress.[1]

4.3.1 Participants & Study Setup

For our study, we recruited 6 professional software developers and 11 PhD students with a major in computer science. Professional developers were recruited from two different software development companies in Switzerland. PhD students were recruited from the University of Zurich. The 17 participants (16 male, 1 female) ranged in age from 20 to 51 years and had an average professional development experience of 7.1 years ($\pm$ 6.7), ranging from 1 to 29 years.

Figure 4.1 depicts the study setup. For this study, we used three different sensors: an off-the-shelf Neurosky MindBand EEG sensor (`http://neurosky.com/`), an Empatica E3 wrist band (`https://www.empatica.com/`), and the Eye Tribe eye tracking device (`https://theeyetribe.com/`). The study took place in a quiet room. Study participants had to wear the EEG headband and the Empatica wrist band and were placed in front of a standard 1920 x 1080 24-inch screen with the eye tracker located in front of the screen.

4.3.2 Study Method

Subjects were first asked to put on the Empatica and the EEG sensors. We then ensured that the devices were connected, data was recorded properly and that the clocks of all recording devices were in sync. Before starting with the actual study, we instructed participants on the procedure and on how to rate their emotions and progress. Prior to starting on the first change task and before switching to the second change task, we asked participants to relax and watch a calming video of fish swimming in a fish tank for two minutes. In our previous study we saw that these two-minute videos relaxed participants and allowed their biometric features drop back to a baseline after about a minute [Fritz et al., 2014a]. After watching the fish tank video, participants started to work on one

[1] A replication package of the study is available at `http://seal.ifi.uzh.ch/people/mueller/SensingDevelopersEmotions`.

Figure 4.1: Study setup with a subject in front of the eye tracker and computer screen, wearing the EEG headband and the Empatica wrist band.

of the two change tasks. The order of the tasks was randomly assigned to each participant, but counterbalanced across all participants.

During their work on the change tasks, we interrupted participants either after they had been working for 5 minutes uninterrupted, or when they showed signs of strong negative or positive emotions, such as cursing or smiling. We chose a time frame of 5 minutes since previous studies found that developers switch tasks on average every 4.5 minutes during their work [Meyer et al., 2014]. During each interruption, we asked participants to rate their emotional state at the moment of the interruption and their current perceived progress. For rating emotions, we followed Russell's 2-dimensional Circumplex model [Russell, 1980] and asked participants to rate them along two axes, a horizontal one for valence and a vertical one for arousal. Based on related work [Mark et al., 2014], both axes ranged from -200 (low) to +200 (high). To measure the perceived progress, we asked participants to rate it on a 5-point Likert scale. Table 4.1 lists the

questions and answer ranges. In addition to these ratings, we asked participants about the reasons for their current state of emotions and progress and what could help them to feel better or make more progress. After working on the first change task for 30 minutes, we stopped participants, had them watch a fish tank video and then had them start working on the second task. For the second task we again followed the same protocol as for the first one.

Table 4.1: Questions and answer ranges during our study.

1. **Please rate how you felt right at the moment of the interruption.** [-200 (very unpleasant) to +200 (very pleasant)]
2. **Please rate how you felt right at the moment of the interruption.** [-200 (very calm/relaxed) to +200 (very excited/stimulated)]
3. **How do you rate your progress right before you were interrupted?** [Likert scale with 1 (completely stuck / no progress at all), 3 (neutral), 5 (in flow / a lot of progress)]

After participants had been working on the second task for 30 minutes, we stopped them, showed them a two-minute fish tank video and then presented them two sets of pictures that are known for inducing specific emotions: one set inducing positive emotions and one inducing negative emotions [Dan-Glauser and Scherer, 2011]. The order of the two sets was randomly assigned to each participant, but counterbalanced across all participants. After each set of pictures, we asked participants to again rate their emotions. We used these picture sets to capture baselines of emotional reactions for each study participant. The picture sets were shown at the end of the study to ensure that they did not influence developers' emotions during their work on the change tasks. In between the two sets, we asked participants to relax and watch a fish tank video.

Once a study participant completed the last assessment of emotions, we stopped the recording of the biometric data and removed all sensors. Then we asked the participant to complete a questionnaire on the demographic background and conducted a brief interview. In the interviews we asked participants when

and why they experience negative and positive emotions during change tasks and which practices they employ to avoid particularly negative emotions. We took hand written notes and audio recorded the interviews.

4.3.3 Change Tasks

Study participants were asked to work on two change tasks for which we provided short descriptions. One task was to write a small Java program that interacts with the StackExchange API [StackExchange, 2014] to retrieve all answers posted by a specific user on StackOverflow and sum up the scores the user earned for these answers. The other task was to implement a new feature in JHotDraw [JHotDraw, 2014], an open-source Java GUI framework. JHotDraw provides a functionality to undo the latest command. For the study, participants were asked to implement a feature that allows users to undo more than one command at once by choosing from a history view of commands.

We chose these two tasks, since they are representative of general change tasks as well as they are not too easy to solve and thus could stress both negative and positive emotions. We ran a pilot study with two subjects working on these two tasks, validating that they can trigger both positive and negative emotions. During the study, participants were allowed to use the Internet and search for help as they would normally do.

4.3.4 Data Collection

During the course of the study we collected biometric measurements that research has previously linked to negative and positive emotions: electro-dermal activity (EDA), electro-encephalography (EEG), skin temperature, heart rate, blood volume pulse (BVP) and various eye-related measurements, such as pupil size. We used a Neurosky MindBand sensor to capture EEG data, an Empatica E3 wrist band to record skin- and heart-related signals, and an Eye Tribe eye tracker to capture eye-related measures. Table 4.2 presents an overview of the captured measurements and the linked emotional aspects.

Table 4.2: Overview of biometric measures and emotion-related aspects previously linked in literature.

Measure	Previously linked to
Eye-related	
Pupil size	excitement [Muldner et al., 2010]; positive and negative affect [Muldner et al., 2009]
Fixations	valence [Carniglia et al., 2012]
Brain-related	
Eye blinks	frustration [Kapoor et al., 2007]; stress and anxiety [Doehring, 1957]
Frequency bands	valence [Sammler et al., 2007, Reuderink et al., 2013]; arousal [Reuderink et al., 2013]; happiness and sadness [Li and Lu, 2009]; various emotions [Murugappan et al., 2008], [Murugappan et al., 2009]
Ratios of frequency bands	valence and arousal [Lin et al., 2010]
Attention and Meditation	valence and arousal [Yoon et al., 2013]
Skin-related	
Electro-dermal activity (EDA)	valence and arousal [Haag et al., 2004], [McDuff et al., 2012]; engagement [McDuff et al., 2012]; frustration [Scheirer et al., 2002], [Freeman, 1940], [Kapoor et al., 2007], [Mandryk et al., 2006]; positive and negative affect [Leite et al., 2013], [Drachen et al., 2010]; various emotions [Picard et al., 2001], [Chanel et al., 2008], [Ekman et al., 1983], [Wagner et al., 2005]
Skin temperature	valence and arousal [Haag et al., 2004]; boredom, engagement and anxiety [Chanel et al., 2008]; various emotions [Ekman et al., 1983]
Heart-related	
Blood volume pulse (BVP)	frustration [Scheirer et al., 2002]; various emotions [Picard et al., 2001]; valence and arousal [Haag et al., 2004];
Heart rate variability (HRV)	anxiety [Rani et al., 2004]; various emotional states [McCraty and Tomasino, 2006]
Heart rate (HR)	valence [Sammler et al., 2007], [Haag et al., 2004]; arousal [Haag et al., 2004]; positive and negative affect [Drachen et al., 2010]; happiness [Steptoe et al., 2005]; various emotions [Wagner et al., 2005], [McCraty and Tomasino, 2006], [Chanel et al., 2008]

Over all study participants, we collected 213 emotion and 213 progress ratings with an average of 12.5 (± 0.9) per participant, ranging from 11 to 14 ratings (as listed in the "Emotions" column of Table 4.7) and an interruption every 4.1 minutes on average, ranging from 0.7 to 5 minutes.

In addition, we collected two emotion ratings per participant for the two sets of emotion inducing pictures. Finally, we also collected answers on the questions we asked for each of the 213 times we interrupted them and for the questionnaires and the interviews at the end.

4.4 Analysis and Results

In this section, we address our three research questions by presenting the analysis and results of the quantitative and qualitative data gathered in our study. We present results aggregated over all participants. In a performance analysis between the group of professional developers and the group of PhD students we only found minor and statistically non significant differences. For the analysis, we divided each task into five subtasks and assessed for each participant the completion of each subtask as fully, partially or not at all completed. Professional developers fully or partially completed a mean of 6.5 of the 10 subtasks (3.5 fully, 3.0 partially), while PhD students fully or partially completed a mean of 6.4 subtasks (3.8 fully, 2.6 partially).

4.4.1 Experienced Range of Emotions & Progress (RQ1)

To examine the range of emotions developers experience during their work on a change task and whether the range of emotions is similar to the one experienced in other situations, we analyzed participants' ratings of arousal and valence during change tasks as well as the ones gathered for the emotion inducing pictures. Figure 4.2 illustrates the range of emotions with respect to the valence and arousal dimension. Blue markers indicate ratings during change tasks and red markers indicate ratings for the pictures that induce negative (low valence) and positive emotions (high valence). While working on the change tasks, each

developer experienced a broad range of emotions, both for the valence and the arousal dimension. Valence ranged for developers from -183 to 200 and arousal from -151 to 181 with an average interquartile range per developer of 65.9 (±49.0) for the valence and 49.8 (±40.0) for the arousal dimension.

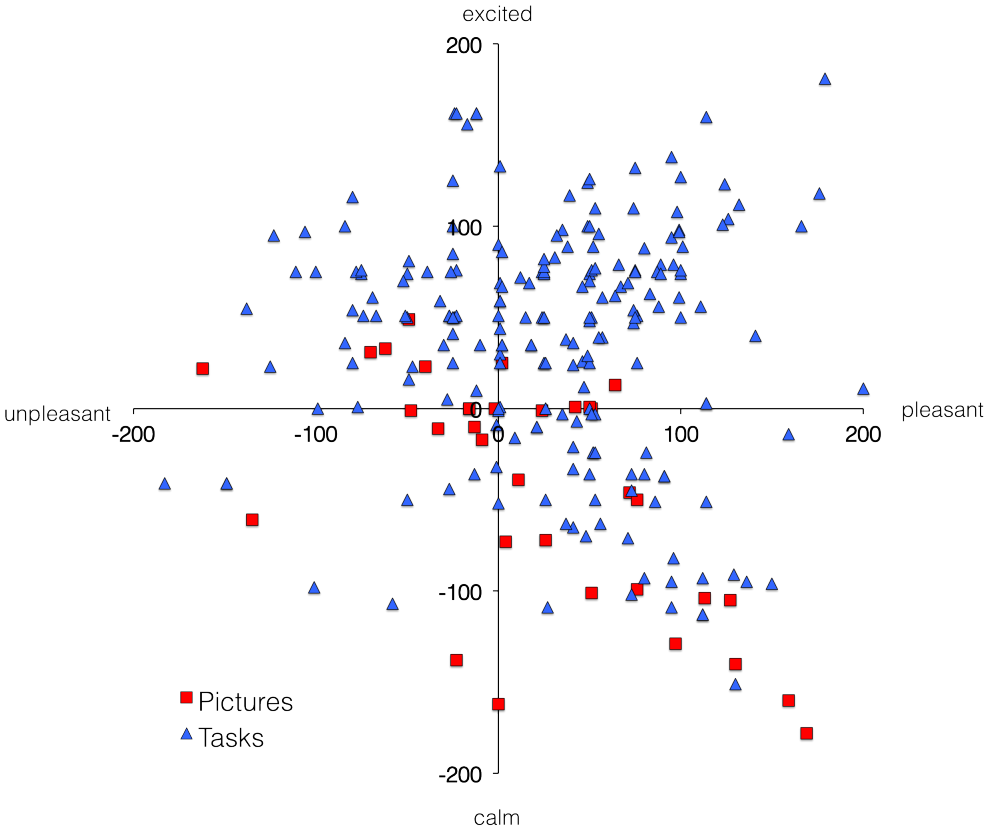


Figure 4.2: Participants' emotion ratings on valence (x-axis) and arousal dimension (y-axis) during change tasks (▲) and after looking at emotion inducing pictures (■).

Since we are particularly interested in positive and negative emotions which are represented by the valence dimension, we compared the valence ratings for change tasks with the ones for the pictures inducing positive and negative emotions. Figure 4.3 depicts box plots of the valence ratings for change tasks

as well as the two picture sets. The box plots highlight that the picture sets generally induced negative and positive emotions and that the range of emotions during change tasks and for the picture sets strongly overlap. These results show that *the emotions developers experience during change tasks cover a broad range of positive and negative emotions and that they are similar to the ones experienced in other situations.*

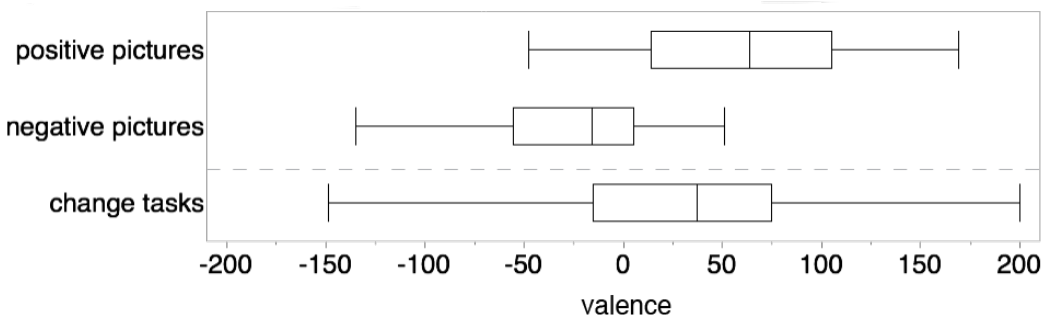


Figure 4.3: Box plots of valence ratings after looking at positive / negative emotions inducing pictures, and during change tasks.

Table 4.3: Progress ratings for the two change tasks.

	stuck		neutral		in flow
	1	2	3	4	5
Task 1	13	17	23	44	9
Task 2	18	31	29	28	1

During their work on each of the two change task, developers also experienced the whole range of progress, from 1 (being stuck) to 5 (in flow / a lot of progress), with a median of 3 and an interquartile range of 2 (see Table 4.3). To examine whether emotion and progress ratings correlate and one might be able to use emotions as a proxy for a developer's progress, we applied a linear mixed model approach to the gathered data. We used a linear mixed model approach instead of other regression models, since research has shown that it is well suited for

repeated measures from the same individual and is able to account for random effects, such as the task or the time of measurement [Gueorguieva and Krystal, 2004]. We defined the self-reported progress rating as the dependent variable. Studies have shown that ordinal data, such as Likert scale ratings, can be used in these kind of parametric tests (*e.g.*, [Norman, 2010]).

Furthermore, we defined the valence and arousal as well as their interaction with the measurement time as fixed effects and the participant, the task and the measurement time itself as random effects. Thereby, we standardized valence and arousal ratings for each participant to accommodate for individual differences in rating. Checking this model against the null model without any fixed effects results in a significant difference ($\chi^2(5) = 106.69, p < 0.001$). This difference shows that the valence and arousal dimensions have a significant effect on the progress in our model.

Table 4.4: Fixed-effects estimates on progress (* indicates significant estimates at the 0.05 confidence level).

Effect	**Estimate**	**Upper p-value** (207 df)	**Lower p-value** (181 df)	**Deviance expl.** (%)
Valence	0.66 (*)	0.00	0.00	28.03
Arousal	0.10 (*)	0.02	0.02	1.09
Time	0.01	0.66	0.66	0.04
Valence:Time	-0.01	0.78	0.78	0.02
Arousal:Time	0.01	0.78	0.78	0.02

Table 4.4 provides an overview of the fixed-effects estimates as well as the upper- and lower-bound p-values for assessing significance. The results show that, at the 0.05 confidence level, both arousal and valence are correlated with progress, however, the correlation between arousal and progress is only very weak. The valence dimension holds by far the highest explanatory power (28.03%) of the whole model (29.12%). The random effects for the measurement time is estimated to be 0, for the task it is in the range [-0.16, 0.16] and the random participant effect is in the range [-1.22, 0.80]. These results indicate that for the random effects that we modeled in our approach, the measurement time has no

effect, the task has a medium effect and the participant has the highest effect on our model.

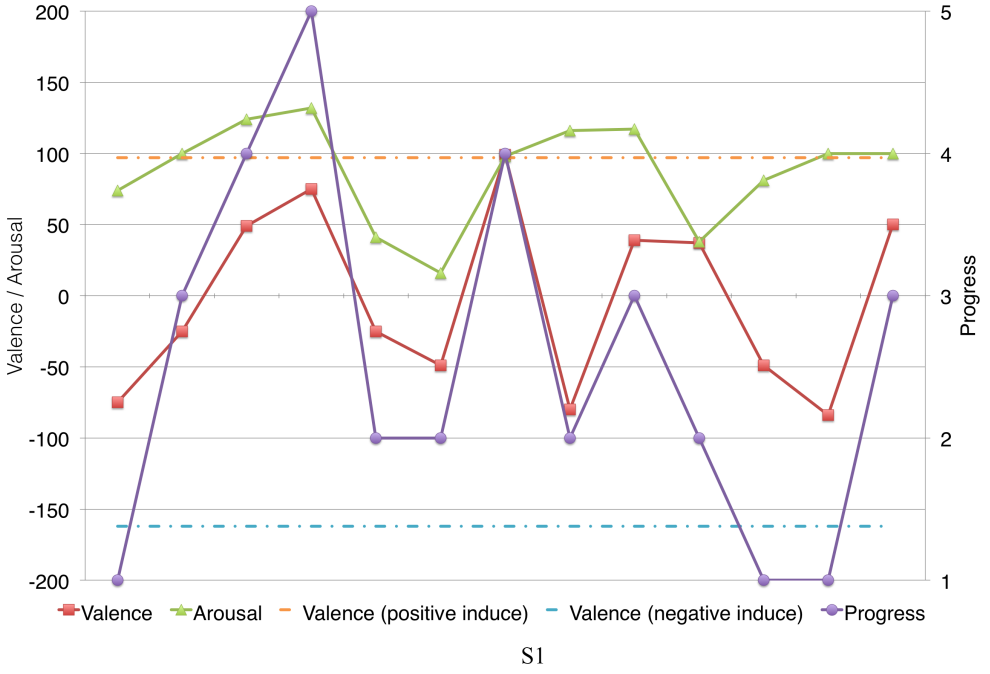
When analyzing individual ratings for each participant, we noticed that for some subjects the valence dimension of their emotions strongly correlated with the progress rating, while for other subjects, it did not. When calculating correlations on an individual basis, we found significant correlations for 12 subjects, but not for the other 5 subjects (S6, S7, S8, S14 and S16). Figure 4.4 provides examples for each of the two groups, with a strong correlation between valence and progress for subject S1 (Spearman's rank correlation coefficient $\rho = 0.88, p < 0.001$) and no significant correlation for subject S6 ($\rho = 0.38, p = 0.28$).

In summary, the results provide evidence that *valence is highly correlated with perceived progress* and might be a good indicator for progress overall, but a lot better for some individuals than others. These findings also support and confirm results on the correlation between emotion and progress found in a study with less participants by Graziotin *et al.* [Graziotin et al., 2013].

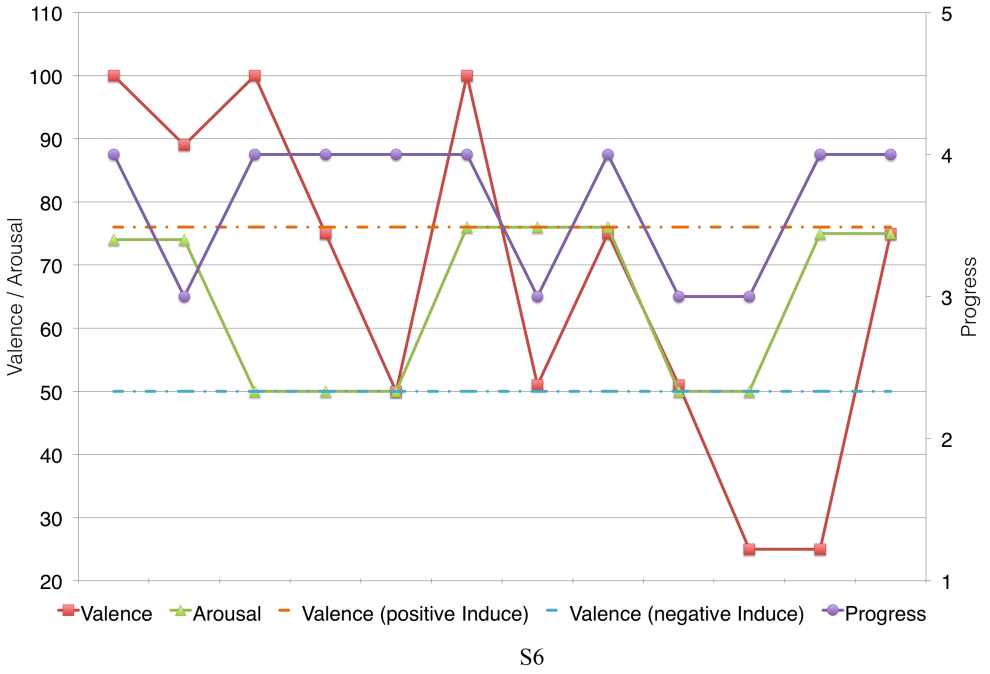
4.4.2 Aspects and Practices Affecting Emotions and Progress (RQ2)

To explore the aspects that affect emotions and progress during a change task and the practices developers employ to avoid negative emotions or a lack of progress, we analyzed the participants' answers to our questions during and after the tasks. We gathered a total of 186 answers out of the 213 data points in which participants' valence, progress or both ratings changed with respect to participants' previous rating during the change task. Since neither the participant's valence or progress changed for the other 27 data points, we did not include them in this analysis.

In 91 of these 186 cases the valence and/or progress increased and in 95 cases they decreased. Based on grounded theory techniques [Martin and Turner, 1986], we used a combination of open, axial and selective coding to identify codes, group them into concepts and categories and find quotes related to the main categories. To avoid observer bias, both authors of this paper performed the coding, discussed and integrated the results.



S1



S6

Figure 4.4: Emotion and progress ratings for S1 and S6.

Table 4.5: Top 5 reasons for a change in emotions/progress.

Increase in emotions/progress	# Cases	# Subjects
localize relevant code	21 (11.3%)	14 (82.4%)
(better) understand parts of the code	18 (9.7%)	13 (76.5%)
next steps are clear	12 (6.5%)	9 (52.9%)
produce something / write code	9 (4.8%)	6 (35.3%)
have new idea	8 (4.3%)	6 (35.3%)
Decrease in emotions/progress	# Cases	# Subjects
difficulty in understanding how parts of the code/API work	33 (17.7%)	12 (70.6%)
difficulty in localizing relevant code	15 (8.1%)	8 (47.1%)
not being sure about next steps	9 (4.8%)	9 (52.9%)
realize that hypothesis on how code works is wrong	9 (4.8%)	7 (41.2%)
missing / insufficient documentation	3 (1.6%)	3 (17.6%)

Aspects that Affect Emotions & Progress. The top five reasons for an increase or a decrease in emotions or progress that participants mentioned when asked during their work on the tasks are summarized in Table 4.5. In both cases, the ability to locate relevant parts of the code and the understanding of parts of the code are the top most reasons and account for 86 of the total of 186 cases (46.2%). When participants were able to locate a starting point or relevant code, it made them feel better and have a good feeling of their progress, while not finding the relevant code resulted in the opposite:

"I've found a starting point almost immediately. I have a good feeling that I'll make significant progress very soon." (S1)

"It's going too slow. I think it's very cumbersome when so much time is needed to understand the project and find a starting point." (S9)

Similarly, a better understanding of parts of the code or the difficulty in understanding can cause changes in a developer's emotions and her feeling of

progress. While a better understanding of parts of the code can lift a developer's emotions, the lack thereof can lead to annoyance and anger:

"I finally understand what I really need to do. There is light at the end of the tunnel." (S12)

"It's unclear how to use `UndoManager`. That pisses me off." (S1)

One aspect that had a very strong impact on a developer's emotion or perception of progress was the writing of code. Thereby, it was not even important whether the code is correct or not, just the mere fact of producing some visible output lifts the spirit of developers as one participant stated:

"I'm feeling slightly better again, since I've produced something visible. At the moment, it's not so important if it's correct or not. Most importantly, I've produced output. That makes me feel great." (S7)

Common among all answers is that participants talked about emotions and progress simultaneously. In many cases, emotions were either mentioned alongside with a perception of progress, or they were mentioned as one affecting the other, for instance, the lack of progress causing annoyance:

"I finally figured out how to do it. I'm really happy and I'm not feeling completely stuck anymore." (S6)

"I can't make any progress. That's annoying." (S13)

This co-occurrence of comments on emotions and on progress in participants' answers further supports our findings from RQ1, indicating a correlation between a developer's positive and negative emotions and the perceived progress.

Practices Employed to Avoid Negative Emotions and Getting Stuck. Since we are interested in understanding how we can support developers in avoiding negative emotions and being stuck, we also asked about the support one could provide in these cases. Most commonly participants stated that a more complete and detailed documentation (27 cases), a description of the high-level architecture (18 cases) and better code examples (17 cases) would be beneficial to feel better and improve progress. When asked more generally about the practices participants employ to avoid negative emotions, three general strategies emerged

from the answers: *switching context when stuck, setting clear goals, and allocating sufficient resources ahead of time.*

Several participants stated that they will switch context and, for example, switch to a different task, talk to others, or take a break. This helps them to feel better and to get new ideas:

"When I'm frustrated, most often I take a coffee break or do something completely different. For example read the [news] online. Just something completely different. Most often I'm more relaxed afterwards." (S9)

"I take a break then and suddenly after the break, the problem is way easier to solve." (S2)

Another practice to avoid negative feelings in the first place, is to set yourself clear goals before starting to work and then actively avoid potential distractions and fade out all other things. Participants thereby also mentioned to give themselves certain rewards for achieving a certain goal:

"What helps me is to set myself a goal. For example, work on this task until then and then, and afterwards, I will give myself some sort of reward, for example, take a break." (S1)

Finally, allocating and planning sufficient resources for a task is a common strategy among participants to avoid negative emotions. Study participants reported that time pressure often leads to stress and frustration for them and they therefore try to reserve enough time for the completion of a task.

4.4.3 Biometric Sensors to Determine Developers' Emotions and Progress (RQ3)

To investigate whether we can use biometric sensors to distinguish between positive and negative emotions as well as episodes of low and high progress that developers experience during change tasks, we applied a machine learning approach to the collected data. Over the course of the participants' work on both change tasks we collected biometric data for a total of 213 intervals. Figure 4.5 illustrates a set of four such intervals for participant S4 together with the collected

EDA and the heart rate signal as well as the participant's emotion and progress ratings. Especially for the EDA signal, the example presented in Figure 4.5 shows a visible difference between the first episode with medium progress and higher valence compared to the last episode with the developer being stuck and a lower valence. Each interval is delimited by our periodic interruptions for which the data is not taken into account. Since emotions typically last for seconds or at the longest up to minutes [Ekman, 1994], we decided to take into account the 10 seconds of biometric data collected before each time we asked a participant to rate her emotions and progress. Due to errors during the data capturing process, we were not able to collect or use heart- and skin-related measurements for S3 and eye-related measurements for S8 in our analysis. All other measurements for these two participants were, however, included in the analysis.

Data Cleaning and Feature Extraction. Since biometric data can be intensely noisy, we applied various noise cleaning steps to the data before extracting features. For the eye-tracking data, we deleted all measurements that were marked as invalid by the eye-tracking device. We also standardized the pupil sizes by participant to account for the differences between participants. Based on research that has shown that pupil size and fixation duration is affected by positive and negative emotions (*e.g.*, [Carniglia et al., 2012], [Muldner et al., 2010], [Muldner et al., 2009]), we extracted various features for fixation duration and pupil size.

The EEG sensor captures the electrical activity of the brain, measured on the scalp. Research has shown that frequency bands, extracted from brain waves, can be used to distinguish between various emotions (*e.g.*, [Li and Lu, 2009], [Murugappan et al., 2008], [Murugappan et al., 2009]). We therefore extracted common frequency bands [Handy, 2005]: α (8-12Hz), β (12-30Hz), γ (30-80Hz), δ (0-4Hz), θ (4-8Hz) from the brain wave signal and also calculated the fraction of each band with one another. Additionally, the Neurosky Mindband sensor provides two pre-processed signals, called Attention and Meditation, that we also used for our analysis. Finally, we extracted the eye blink rate from the EEG signal using a method proposed by Manoilov [Manoilov, 2007]. We could have extracted the eye blink rate from the eye tracker data. However, since we were not able to capture this data for S8, we used the EEG signal.

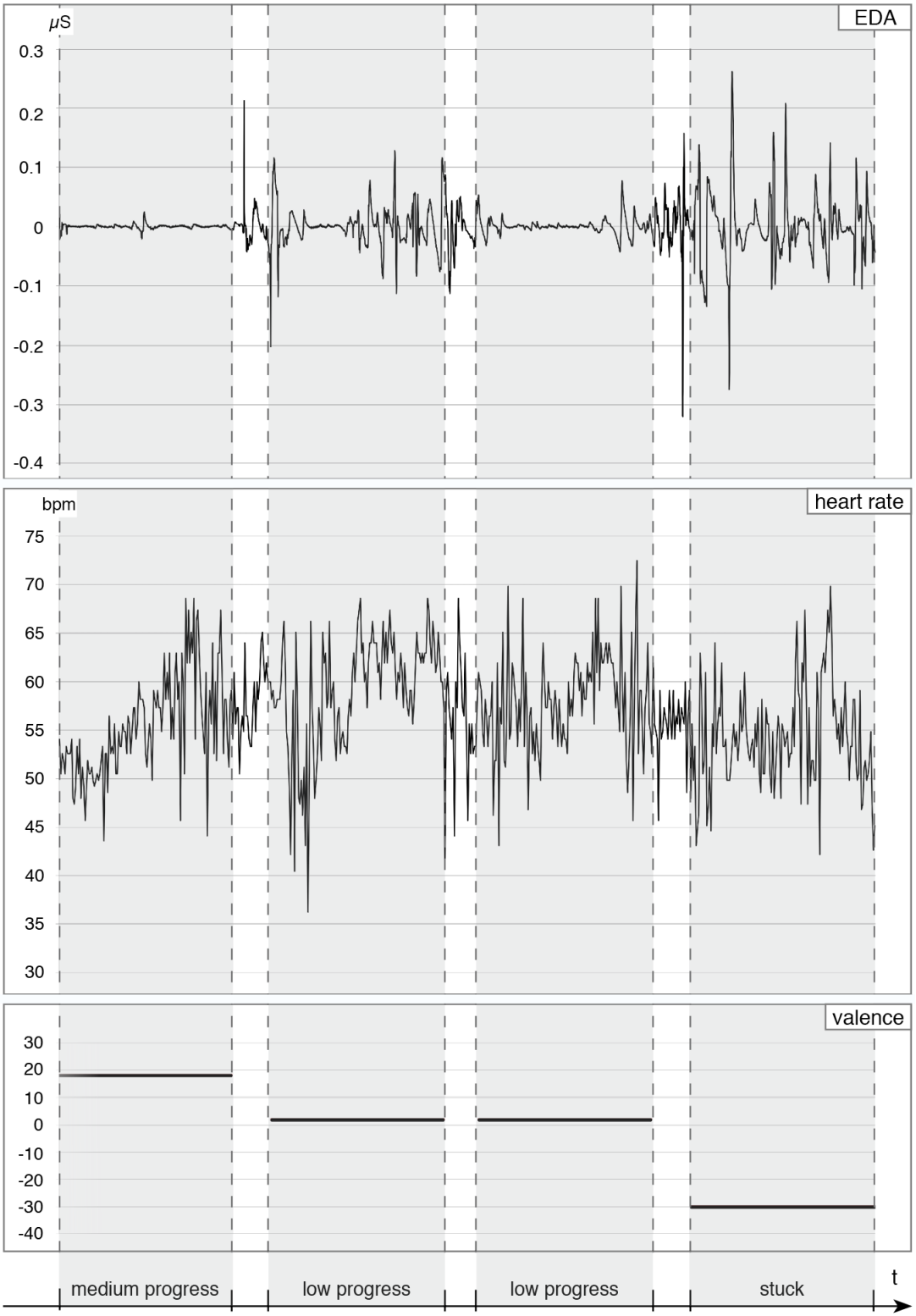


Figure 4.5: Exemplary biometric data, emotion and progress ratings collected over 4 intervals for participant S4.

The EDA signal consists of two parts: the low frequency, slowly changing tonic part, and the high frequency, fast changing phasic part [Schmidt and Walach, 2000]. We used a low-pass and a high-pass Butterworth filter to extract the phasic and tonic part from the EDA signal. In particular features related to the peaks in the phasic signal, but also features extracted from the tonic part of the signal, were closely linked to emotions in previous studies (*e.g.*, [Drachen et al., 2010], [Leite et al., 2013], [Picard et al., 2001]). The Empatica E3 sensor also measures skin temperature that research has used to infer emotional states (*e.g.*, [Chanel et al., 2008], [Ekman et al., 1983], [Haag et al., 2004]). We included these features in our analysis as well.

For the heart-related data we focused on features that describe peaks of the blood volume pulse (BVP) signal [Peper et al., 2007] and we also extracted the mean heart rate that was used in research to asses emotions [Sammler et al., 2007], [Steptoe et al., 2005]. By a simple transformation, the heart rate can be used to calculate the heart rate variability (HRV) that represents the variation in the time interval between two consecutive heart beats. In research, HRV was previously used to infer various emotional states [Mietus et al., 2002] and is usually analyzed by calculating the mean and the standard deviation of the time between two successive heart beats [Bilchick and Berger, 2006]. We added these features to our analysis as well.

Data Labelling. To distinguish between positive and negative emotions, we focused on the valence dimension of the emotion ratings. Given the individual differences in the way participants rated their emotions, we used the emotion inducing pictures, known for inducing particularly negative and positive emotions [Dan-Glauser and Scherer, 2011], to determine a baseline for each participant. We calculated the mean of the valence ratings for the positive and negative emotion inducing pictures and then labelled the ratings from the change tasks below the mean as negative and the ones above the mean as positive. We manually inspected all valence ratings for each participant to disambiguate in cases the ratings were very close to the mean. Only for 5 cases out of the 213, the ratings were almost identical with the mean. In these five cases we additionally took the

participants' comments into account to unambiguously label them. We ended up with 128 ratings for positive and 85 ratings for negative emotions.

To distinguish between low and high progress, we used participants progress ratings on the 5-point Likert scale (1 = completely stuck / no progress at all, 3 = neutral, 5 = in flow / a lot of progress) and classified ratings of 1 and 2 as low progress and ratings of 4 and 5 as high progress. Since we were not interested in episodes where subjects reported their progress as neutral, we removed these instances from our analysis. We ended up with 79 instances of low progress and 82 instances of high progress.

Machine Learning. For our machine learning classifier, we used the Java-based framework Weka [Hall et al., 2009]. We first partitioned our data by participant and task. Since each participant worked on two different change tasks and we collected 5 to 7 emotion and progress ratings per task and participant, we ended up with 17 times 2 (=34) participant-task combinations, each having 5 to 7 data points. We partitioned the data by participant and task to avoid having data points from the same participant and the same task in both the training and testing set. We then used a leave-one-out method and trained our classifier in turn with all participant-task combinations except one, and used the remaining combination as test set. For feature selection we used ConsistencySubsetEval, a Weka implementation of an algorithm that chooses a feature subset based on the consistency between the data [Liu and Setiono, 1996]. As classifier, we opted for a decision tree classifier and used J48, the Weka implementation of C4.5 [Quinlan, 1993]. We used a decision tree classifier under the assumption that its non-parametric characteristics [Maimon and Rokach, 2006] would fit our collected data, which often exhibited a non-parametric distribution.

Results. Table 4.6 presents the results of our machine learning classification. When classifying emotions into positive and negative ones, a classifier trained on biometric data is able to predict 71.36% of all cases correctly. Compared to a naive predictor that always predicts the most dominant class but never any other class, this is an improvement of 18.76%. Compared to a random predictor that randomly predicts one of the two classes, this is an improvement of 42.72%. The features with the most predictive power for this kind of prediction were the

brainwave frequency bands, the pupil size, as well as the heart rate. Predicting the progress achieved similar accuracy. Our machine learning approach was able to classify 109 out of 161 cases correctly (67.70%). This is an improvement of 32.93% compared to a naive predictor and 35.40% compared to a random predictor. To classify progress, the EDA tonic signal, the temperature, brainwave frequency bands, and the pupil size were most predictive. These results indicate that we can use biometric measurements to distinguish fairly accurately between positive and negative emotions that a developer experiences during a programming task. Slightly less accurate, but still better than a naive or random predictor, it is also possible to distinguish between episodes of low and high perceived progress. The results also indicate that a combination of multiple sensors works best for these kind of predictions.

To examine the results in more detail and by participant, Table 4.7 lists all results partitioned by participant. The results show that for some participants, *e.g.*, S11 or S14, the prediction of positive and negative emotions as well as low and high progress works very well, while for other participants, such as S16 or S17, both predictions do not achieve great accuracy.

Finally, we also examined if we could train a classifier and then use it for classifying emotions and progress of a participant that the classifier was not previously trained on. Therefore, we trained the classifier in turn for all participants except one and used the data of the remaining participant for testing. While the accuracy for distinguishing positive and negative emotions is identical (71.36%), the accuracy to distinguish between low and high progress is slightly lower (63.35%).

4.5 Discussion

Individual Differences. While our results show that over all developers there is a correlation between emotions and perceived progress and biometric features can be used to predict emotions and progress, the analysis also shows that there are strong individual differences with respect to the correlation and the classification. For instance, while the machine learning classifier for emotions

Table 4.6: Machine learning results for classifying emotions and progress together with the features selected for each classifier (Δ represents the difference to the baseline).

Prediction	Correct	Precision	Recall	Selected features
Emotion	71.36%	64.32%	82.03%	ΔAlpha, ΔBeta/Theta MinPupilSize, ΔMeanHR
Progress	67.70%	67.85%	68.29%	ΔAlpha, ΔBeta/Theta ΔMeanTempPeakAmpl, MaxPupilSize, ΔMeanPupilSize, ΔMeanSCL

is only correct in 30.77% of the cases for participant S7, it goes up to 85.71% of correct cases for S3 (see Table 4.7). Khan *et al.* [Khan et al., 2013] already pointed out that, due to the widely varying range and perception of emotions by individual participants, it is very difficult to find relationships between an individual's emotions and other aspects, such as progress, that hold for more than just a few people. Especially given the high variability in biometric measures between individuals [Mandryk, 2008], an approach that more specifically takes into account individual differences and is, for instance, trained specifically for a developer, might greatly improve our results. While this has the disadvantage of requiring training sessions for each user, we assume this will be negligible compared to the potential benefit it could bring.

Developer Support. The quantitative and qualitative results of our study also contribute new knowledge on how to support a developer and ensure that her time is spent as productive as possible. Since developers frequently had negative emotions and experienced little progress when they did not understand parts of the code, one could use an approach with a classifier on emotions and progress to identify places in the code that are particularly difficult to understand and might benefit from a code review or refactoring. In particular, given the new advances in eye tracking technology that make them a lot more affordable and easy to add to any existing setup, one might be able to use the classifier to track the difficult parts of the code on the level of code lines. Thus, such a classifier might be used as a new kind of code smell detector that could more automatically add a

Table 4.7: Machine learning results partitioned by participant.

Participant	Emotions			Progress		
	Correct	Total	% Correct	Correct	Total	% Correct
S1	8	13	61.54	3	10	30.00
S2	11	14	78.57	6	10	60.00
S3	12	14	85.71	7	12	58.33
S4	8	11	72.73	6	7	85.71
S5	11	13	84.62	6	10	60.00
S6	8	11	72.73	4	5	80.00
S7	4	13	30.77	8	10	80.00
S8	10	13	76.92	4	7	57.14
S9	11	13	84.62	6	9	66.67
S10	9	13	69.23	8	11	72.73
S11	10	12	83.33	9	9	100.00
S12	10	12	83.33	4	9	44.44
S13	10	13	76.92	6	10	60.00
S14	10	12	83.33	9	12	75.00
S15	6	12	50.00	11	11	100.00
S16	7	12	58.33	5	8	62.50
S17	7	12	58.33	7	11	63.64
Total	**152**	**213**	**71.36**	**109**	**161**	**67.70**

human aspect to code analysis. Furthermore, when a developer is trying to locate relevant code, a classifier on emotions and progress could be used to determine the times when code recommendations would be particularly helpful. This would allow to avoid overwhelming developers with continuous recommendations but provide them at opportune moments when the developer is most susceptible to them. In addition, knowing when a developer has particularly negative emotions or getting stuck one could recommend taking a break, while in a state of flow, tool support could be built to avoid interruptions through notifications or coworkers.

Negative Emotions. While results have shown that negative emotions are often correlated with low progress, it is important to note that avoiding negative emotions at every cost will not always automatically lead to more progress. In certain situations, negative emotions might actually be a necessary part to solve a problem and lead to higher progress later on, and occasionally being frustrated by a task might provide indirect benefits. As Wrobel *et al.* [Wrobel, 2013] already

observed in their study, negative emotions can act as an activator for developers to become more productive. Future studies are needed to explore this positive aspect of negative emotions further and determine if there are ways to distinguish between possibly beneficial or detrimental experiences of negative emotions, such as frustration.

Ethical and Privacy Concerns. Finally, with the introduction of biometric sensors there are also ethical and privacy concerns to be addressed. While advances in sensor technology might decrease the physical invasiveness of these sensors, the capturing of huge amounts of very personal data can raise several ethical and privacy concerns in a developer. By focusing on the individual and providing personalized support to the developer without sharing the fine-grained and very developer-specific data with others, we can help to assuage these concerns. In future studies, we plan to further investigate the impact of such support on the individual and how we might be able to avoid such concerns.

4.6 Threats To Validity

External Validity. Since participants only worked on two change tasks, the generalizability of our study might be limited. We tried to mitigate this risk by carefully choosing study tasks representative of typical change tasks, either requiring the use of a popular API or requesting a change in a system commonly used for studies. Another threat to generalizability is the selection of participants. We tried to limit this by recruiting participants with various backgrounds.

Internal Validity. We observed participants while they were working in our lab study setup. In particular the environment might trigger different emotions or progress than participants would usually experience in their work environment. We tried to mitigate this risk by selecting representative tasks that triggered a broad range of emotions and validated it with emotion inducing pictures. Future work needs to investigate if these results can be ported to life work environments.

During the study, we regularly interrupted participants and asked them to rate their emotions and progress. These interruptions might have influenced

participants' performance and ratings. We tried to mitigate this risk by choosing a time interval between interruptions that is representative of the time interval of developers' task switches.

Construct Validity. As one part of this study, we used biometric measurements to predict the positive and negative emotions as well as developers' perceived progress. The data captured with these sensors might be affected not only by the emotions and progress that developers experienced during the study tasks, but also by study participants' personality traits or their general stress level. To mitigate this risk, we conducted the study in a quiet environment and limited all unnecessary distractions. Additionally, we periodically let the study participants watch a calming and relaxing video and collected biometric baseline data that we used in the analysis to compare the data collected during the study tasks.

4.7 Conclusion

Software developers experience a broad range of emotions during their work. Previous studies have shown that biometric sensors can be used to distinguish between positive and negative emotions. These studies focused on certain kinds of tasks, such as very small analytical tasks, that are not representative of development tasks. In the presented research, we built upon and extend previous work to the context of software change tasks and the classification of a developer's progress as well as emotions. The results of our study show that using machine learning, we are able to distinguish between positive and negative emotions in 71.36% of all cases and between low and high progress in 67.70%. Our results also show that emotions and perceived progress are highly correlated and illustrate aspects and practices that affect emotions and progress. These insights provide a lot of opportunities for future work that could have direct potential impact on a developer's work and productivity. One could, for instance, provide automatic support to a developer by recommending code examples, relevant documentation or even just a short break when the developer is getting stuck and frustrated and it might be most beneficial. Similarly, one can imagine tool support to minimize and postpone interruptions when the developer is in flow and the cost of an

interruption would be particularly high. In future work, we intend to explore these opportunities and further examine more individualized classifiers that take into account differences in people's biometric data and thus might be able to determine a developer's emotions and progress even better.

Acknowledgments

We thank all study participants for participating in our study and Elaine Huang, Gail Murphy and the reviewers for their valuable feedback.

5

iTrace: Enabling Eye Tracking on Software Artifacts Within the IDE to Support Software Engineering Tasks

Timothy Shaffer, Jenna Wise, Braden Walters, Michael Falcone,
***Sebastian C. Müller**, and Bonita Sharif.*
Published at the International Symposium on Foundations of Software Engineering (Tool Demonstrations Track), 2015
Contribution: Tool implementation and paper writing

Abstract

The paper presents *iTrace*, an Eclipse plugin that implicitly records developers' eye movements while they work on change tasks. *iTrace* is the first eye tracking

environment that makes it possible for researchers to conduct eye tracking studies on large software systems. An overview of the design and architecture is presented along with features and usage scenarios. *iTrace* is designed to support a variety of eye trackers. The design is flexible enough to record eye movements on various types of software artifacts (Java code, text/html/xml documents, diagrams), as well as IDE user interface elements. The plugin has been successfully used for software traceability tasks and program comprehension tasks. *iTrace* is also applicable to other tasks such as code summarization and code recommendations based on developer eye movements. A short video demonstration is available at `https://youtu.be/3OUnLCX4dXo`.

5.1 Introduction

The use of eye trackers has become increasingly popular in the software engineering (SE) community, as evidenced by the increasing number of publications in mainstream conferences and journals [Fritz et al., 2014a], [Busjahn et al., 2015], [Rodeghero et al., 2014], [Müller and Fritz, 2015b]. Eye trackers have also become cheaper and more affordable. An eye tracker allows an SE researcher to collect eye movements of software engineers while they work on software tasks such as adding a new feature, fixing a bug, or on a general comprehension task. The eye movement data is used to study the cognitive thought processes [Rayner, 1998] of developers as they perform a task using different software artifacts. Existing eye-tracking studies have mainly studied developers comprehending software artifacts such as source code, models such as UML diagrams, and software visualizations.

An eye tracker consists of both hardware and software. The hardware is a physical device that usually sits under the monitor. The software provided by eye tracking vendors is in the form of an experiment builder (*e.g.*, Tobii Studio from Tobii Inc.) that allows researchers to build the experiment workflow using various stimuli such as a still image, a website, a video recording, or free form desktop recording.

Most existing eye tracking studies (besides [Walters et al., 2014] and [Kevic et al., 2015]) done in the SE community use small snippets of code shown as an image to study participants. An ad hoc system to support scrolling using slider bar events for a few small programs was done by Uwano *et al.* [Uwano et al., 2006] but the tool is unavailable. In all other studies, the image needs to be displayed on the screen all at once and participants are not allowed to scroll as scrolling would interfere with data collected and make post processing extremely difficult if not impossible. This is because the eye tracker is not aware of the type of stimulus presented to it. It reports the (x, y) coordinates where a person is looking on the screen, but is not aware of what exists at that position. When the image is kept static and not allowed to move (which happens during scrolling) it is easy to map (after the experiment) what the (x, y) coordinates represent on the stimulus (which would be a snippet of source code in case of program comprehension). This mapping process is not automatic and needs to be done by the researcher after the experiment is conducted by creating areas of interest on the stimulus, which is an extremely tedious process. Some of this difficulty is alleviated by using a tool such as eyeCode[1] that automatically detects words in the stimuli, but even with eyeCode, the post processing is still tedious and only works on a single static image.

The above mentioned setup works well in studies conducted in psychology (where studies mainly consist of reading text paragraphs and looking at images), but does not scale well to SE. In order to run realistic experiments with eye trackers in SE, we need to be able to run eye tracking studies on realistic systems consisting of hundreds of source files and not be limited to a single static view. *iTrace* was designed exactly for this purpose. There is no need for the researcher to manually map (x, y) coordinates to source code elements as all of this time-consuming labor-intensive process is now done automatically by *iTrace*. *iTrace* runs uninterrupted in the background within Eclipse, recording developers' eye movements while they are working. The plugin is open source under the GPL license at `http://seresl.csis.ysu.edu/iTrace`.

[1] https://github.com/synesthesiam/eyecode/

5.2 iTrace Architecture

This section describes a high-level overview of the *iTrace* plugin architecture and its integration into the Eclipse UI. Additional details about architecture design and our initial ideas for the tool are discussed in [Walters et al., 2013]. The current implementation handles gazes on Java source code, text files including HTML and XML files, and Eclipse UI elements. *iTrace* is designed with a modular architecture, and it is easy to write new handlers for different file types to collect fine-grained data at the statement level.

5.2.1 Overview

Enabling eye tracking for the IDE requires implementing three high-level tasks: 1) capturing a user's gazes from the tracking device, 2) determining which UI element the user is looking at within the IDE, and 3) processing this information toward some functional goal. Each of these tasks must be done in parallel to achieve maximum responsiveness. To overcome this challenge, we use a multithreaded design consisting of a thread for each task that communicates with other threads via shared blocking queues. Our architecture makes use of a `Gaze` object class and three main Java interfaces for working with gazes: `IGazeHandler`, `IGazeResponse`, and `ISolver`. Each is described below.

`Gaze` – Represents position, time, pupil size, validity, and error information for each gaze detected by the eye tracker.

`IGazeHandler` – Describes a handler that accepts a gaze on a widget and returns a `IGazeResponse`.

`IGazeResponse` – Describes information observed by a specific gaze on a specific widget.

`ISolver` – Describes what to do with a specific `IGazeResponse` object.

5.2.2 Integration with Eclipse

To implement our design as a plugin within the Eclipse framework we make use of the `Widget` class (`org.eclipse.swt.widgets.Widget`). This class represents

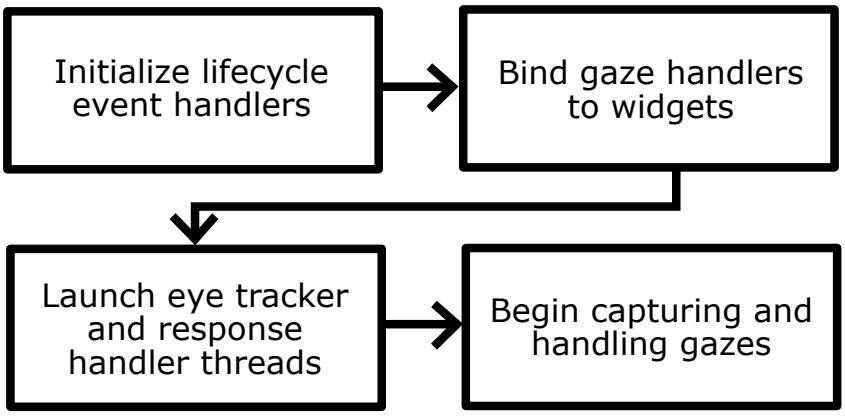


Figure 5.1: Plugin initialization process.

each user interface object that is part of the IDE, and as such stores some content being viewed by the user. It also exposes two methods – `getData()` and `setData()` – which we use to bind and access gaze handler objects such that each UI element with content of interest has its own gaze handler instance. The plugin is initialized following the process summarized in Figure 5.1.

The lifecycle event handlers manage tasks such as pausing gaze processing when the IDE loses focus or is minimized as well as initializing gaze handlers on new editor windows (*e.g.*, when the user opens a new source file). Eye tracking devices are implemented in the plugin by implementing the `IEyeTracker` interface and modifying the eye tracker factory class. We use the Java Native Interface (JNI) to interface with the eye tracker SDK in C or C++ (for *e.g.*, the `TobiiTracker` class). For convenience, we have also implemented the system mouse tracker as a proxy for eye movements when an eye tracker is unavailable for testing.

Capturing and processing gazes follows the process summarized in Figure 5.2 and continues until stopped by the user or paused by minimizing or taking focus from the IDE. If capturing is paused, it will automatically resume when the IDE window is restored or acquires focus. Resizing of the window is also supported by *iTrace*.

Synchronization of thread communication is handled through the Java generic class `LinkedBlockingQueue` used to implement queues shared by threads. Gazes

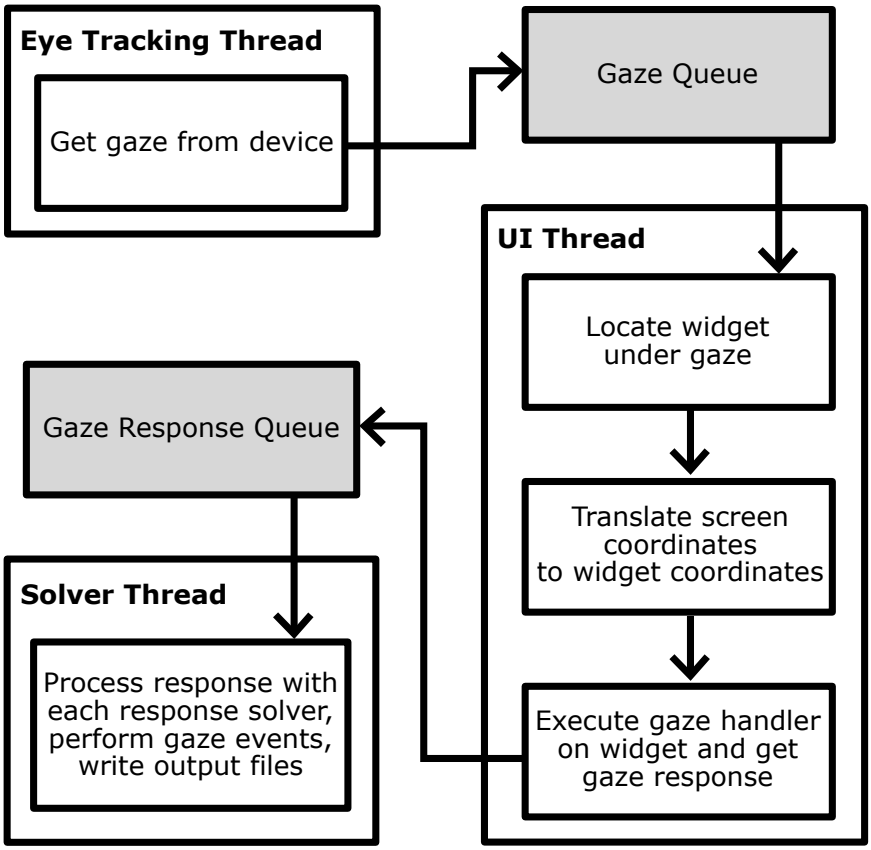


Figure 5.2: Handling of gazes and gaze responses by multiple threads.

are consumed by the UI thread in fixed-size chunks and gaze responses are consumed entirely by the response handler thread in order to reduce the amount of locking that must take place.

Source code tracking as described in Section 5.3 is implemented using the `ASTParser` class built into Eclipse to parse Java syntax and generate an abstract syntax tree (AST) storing line and column numbers of each source code entity. We implement a gaze handler for the editor window that translates gaze coordinates to line and column numbers using methods exposed by the `StyledText` Eclipse class, which stores source code content displayed by the IDE. Using the line and column numbers under the gaze and the line and column numbers of each source code entity we are able to determine the entity that the user is currently viewing.

5.3 Features

In this section, we describe the features that *iTrace* currently provides.

5.3.1 Session Creation

The session information consists of a generated session ID, the session purpose (new feature, bug fix, refactoring, general comprehension or other), and a longer session description. In addition to this, the developer ID with an optional developer name is requested. Session information is required before tracking starts.

5.3.2 Calibration

Every eye tracker needs to be calibrated before use. During calibration, the user is asked to look at several dots that appear on the screen while the eye tracker uses the user's eye features along with its 3D eye model to calculate gaze data. *iTrace* uses a 9-point calibration mechanism, interfacing with the eye tracker's native libraries to calibrate. When calibration is complete, *iTrace* displays the results of the calibration so that the user is able to verify the calibration quality. The user can accept the calibration, or recalibrate.

5.3.3 Displacement Adjustment

In order to check for displacement or drift, *iTrace* supports a crosshair feature. When enabled, it displays a green crosshair showing where the user is looking. For some users, there is always some displacement of the eye gaze as shown by the crosshair vs. the actual point looked at on the screen. We test this by asking the user to look at certain words at the top left, bottom left, top right and bottom right after calibration. If the actual eye gaze is off from the intended position, one is able to adjust the displacement on the x and y axis as needed to bring the crosshair in line with the user's gaze. This displacement setting is used throughout each session. It is not always necessary to adjust for drift and should be used only when required.

5.3.4 Source Code Entity Level Tracking

iTrace supports fine-grained tracking of software artifacts at the line and word level. In particular, emphasis is given to source code as it is the most structured and semantically rich artifact. The current *iTrace* model is able to map gazes to source code entities (SCE) types such as classes, methods, variables, comments, method invocations, conditional expressions, and *enum*, *import*, *for*, *if*, *while*, and *switch* statements. For each of these types, it also states how the SCE type is used in the code, *i.e.*, a declaration vs. an invocation. For example, if a user looked at a method call, it would be considered a `use` of the *method* type.

5.3.5 Raw Data Exports

The number of gazes recorded by *iTrace* depends on the number of samples per second output by the eye tracker. Each gaze recorded by the eye tracker is used to generate a gaze response object. *iTrace* currently supports gaze response export into JSON and XML. An example gaze response in XML format is shown in Figure 5.3. Properties of the gaze itself, such as time, pupil diameter, validation, and (x, y) coordinates, are stored as attributes of the `response` element. Of these properties, all except `system_time` and `nano_time` are read from the tracker. `system_time` is the POSIX time on the host system expressed in milliseconds, and `nano_time` is a high-precision time expressed in nanoseconds. The `type` attribute reports the region of the Eclipse UI where the gaze fell. In the example, the gaze was in a text editor, so the `response` element has some additional attributes, including the filename, line, column, editor font height in points, line height in pixels, and the (x, y) coordinates of the upper left corner of the line. For Java files, the Eclipse AST is queried to determine the source code elements on which the gaze fell. This step is necessary, since relying only on the line number would not be accurate if the content of the file is changed during a tracking session.

In the example, the `sce` elements, sorted from most specific to least specific, report the types of the source code elements, how they are being used, their positions within the file, and the number of characters comprising the source code elements.

In this case the gaze fell on a call to the method `java.lang.String.length()` in the declaration of the `trim` variable, inside an if statement in the `canonPath` method, within the `tasks.MiscUtilities` class.

```
<response line_base_y="412" line_base_x="90" col="23" line="38"
font_height="24" line_height="37"
path="C:\programs\src\tasks\program5.java"
nano_time="2969874330552" system_time="1433180043835"
tracker_time="1433178283244"
right_pupil_diameter="3.151641845703125"
left_pupil_diameter="3.095123291015625" right_validation="1.0"
left_validation="1.0" y="437" x="510" type="text"
name="program5.java">
  - <sces>
        <sce type="METHOD" name="java.lang.String.length()"
            end_col="30" start_col="17" end_line="38" start_line="38"
            total_length="13" how="USE"/>
        <sce type="VARIABLE"
            name="tasks.MiscUtilities.canonPath.trim" end_col="30"
            start_col="10" end_line="38" start_line="38"
            total_length="20" how="DECLARE"/>
        <sce type="IFSTATEMENT" name="IfStatement-l35c4"
            end_col="5" start_col="4" end_line="51" start_line="35"
            total_length="617" how="DECLARE"/>
        <sce type="METHOD" name="tasks.MiscUtilities.canonPath
            (java.lang.String)" end_col="3" start_col="2" end_line="77"
            start_line="17" total_length="2192" how="DECLARE"/>
        <sce type="TYPE" name="tasks.MiscUtilities" end_col="1"
            start_col="0" end_line="78" start_line="15"
            total_length="2270" how="DECLARE"/>
    </sces>
</response>
```

Figure 5.3: An *iTrace* gaze response record.

5.3.6 Fixation Exports

In eye-tracking terminology, a fixation [Rayner, 1998] is when the eye stabilizes for a certain duration at a particular point of interest. *iTrace* calculates fixations by running a basic fixation filter on the raw data. In simple terms, a set of raw

gazes that fall around the same area are grouped and merged together to form a fixation record, along with the fixation start time and the duration.

5.4 Usage Scenarios

The usage scenarios of *iTrace* fall into two broad categories. First, it can be used as a method to assess and learn about how developers navigate and look at different software artifacts. Second, the data can be used to inform software development tasks. An example of *iTrace* being used in each of these two scenarios is described below.

5.4.1 Program Comprehension

iTrace has recently been used by Kevic *et al.* [Kevic et al., 2015] to investigate detailed developer behavior while performing realistic change tasks on a large open source system. The study was conducted on 22 developers. To investigate the added benefit of data generated from *iTrace*, the study compared eye tracking data with Mylyn interaction history data, both of which were gathered simultaneously. The authors found that *iTrace* does capture more contextual data on source code elements, and more importantly captures different aspects of developer activity compared to interaction data.

5.4.2 Software Traceability

The data generated from *iTrace* has also been used by Walters *et al.* [Walters et al., 2014] as input to help recover software traceability links. The links are automatically derived between bug reports and source code entities from a set of developer eye tracking sessions on bug fixes. The concept of collective intelligence (ability to gather knowledge from other developers towards a common goal) [Storey et al., 2014] was used in the above algorithm. The results were very promising, and eye gaze does indeed seem to work well to uncover links between relevant code entities and the bug report. They developed an algorithm to find relevant entities looked at using a weighting scheme based on time. This helps

weed out source code entities that are looked at initially but later abandoned. The version of *iTrace* used in this study is available as a release[2]. The link generation algorithm used on *iTrace* data along with the replication package is also publicly available[3].

5.4.3 Future Scenarios

Besides the above scenarios, there are many software tasks that can directly benefit from the data that *iTrace* provides. Both Rodeghero *et al.* [Rodeghero et al., 2014] and Ali *et al.* [Ali et al., 2012] use eye tracking data as a means to improve automatic code summarization and weighting schemes in software traceability link recovery respectively. Fritz *et al.* [Fritz et al., 2014a] also use eye tracking data to predict task difficulty. However, these studies use small snippets of code that had to fit on one screen. With *iTrace* these types of studies could be conducted on large systems or even larger snippets of code, thereby providing fine grained eye tracking data that is mapped to source code entities.

5.5 Current Limitations

At the time of this writing, two eye trackers, the Tobii X60 and Tobii EyeX are supported. However, the modular architecture of *iTrace* allows for easy addition of new devices. Additionally, the tool is implemented as an Eclipse plugin and therefore cannot capture gazes outside of Eclipse. It can however determine that the Eclipse window is not in focus due to the lack of gazes. Tracking of elements behind an open dialog box such as the search window is not currently supported. Finally, the current version of *iTrace* does not support code folding yet. In an upcoming version of *iTrace*, we plan to support search view tracking and code folding.

[2] `https://github.com/YsuSERESL/iTrace/releases/tag/icpc2014`
[3] `http://www.csis.ysu.edu/~bsharif/itrace-pilot/`

5.6 Conclusions and Future Work

The paper describes *iTrace*: an Eclipse plugin that makes collecting eye gaze on software artifacts possible on large software systems. For source code documents, *iTrace* maps the eye gazes to fine-grained source code entities looked at. The main contributions of this paper are a) an eye-aware Eclipse plugin b) easy to comprehend gaze export format for source code entities and c) demonstrated usage of *iTrace* for program comprehension and software traceability tasks.

Currently, fine-grained line and word level support is provided for Java files. Other files such as text, xml and html files are tracked, but not at the line-level. *iTrace* can also record gazes on Eclipse UI elements, such as the project explorer. Additional handlers can be written for finer-grained information from structured text files. These custom handlers can be specifically tailored to each researcher's needs.

The current research provides several directions for future work. First, support for more eye trackers is needed. Next, support for tracking Javadocs and UML diagrams will be added. Other features, such as support for tracking during search, code folding, replaying gazes over a particular method and keeping track of method renaming are planned.

Acknowledgments

Special thanks to Huzefa Kagdi for inspiring conversations leading to the development of *iTrace*.

6

Leveraging Biometric Data to Boost Software Developer Productivity

Thomas Fritz and ***Sebastian C. Müller***
Published at the International Conference on Software Analysis, Evolution, and Reengineering (Future of Software Engineering Track), 2016
Contribution: Paper writing

Abstract

Producing great software requires great productive developers. Yet, what does it really mean for an individual developer to be productive, and what can we do to best help developers to be productive? To answer these questions, research has traditionally focused on measuring a developer's output and therefore suffered

from two drawbacks: the measures can only be calculated after a developer finished her work and these measures do not account for individual differences between developers.

The recent advances in biometric sensor technology offer new opportunities to measure a developer's cognitive and emotional states in real-time and thus allow us to know more about what an individual developer is currently experiencing and what might foster or impede the developer's productivity. Results from recent research studies demonstrate the potential that biometric data has to accurately predict aspects of a developer's work, such as perceived task and code difficulty, progress and interruptibility of a developer. This opens up new opportunities for better supporting developers in their work and, for instance, prevent bugs from entering the code, reduce costly interruptions, and foster a better and more productive work flow. Our vision is that biometric sensing will be integrated into a developer's work and that biometrics can be used to boost the productivity of each individual developer.

6.1 Introduction

"Software is eating the world." This statement by Marc Andreesen in a Wall Street Journal Op-Ed highlights the fact that software has become the backbone of countless major businesses, a trend that is likely to continue for the foreseeable future [Andreessen, 2011]. Yet, there never seem to be enough software developers to satisfy the demand, despite the immense growth in the number of software developers over the years, with an estimate of eleven million professional developers in 2014 [IDC, 2014]. Beyond simply training more developers, one promising and complementary way to address this demand is to unleash the untapped potential of each "individual" developer. This raises some intriguing unanswered questions: What does it really mean for an individual developer to be productive? How are developers doing their work, what is going on in their minds and when are they experiencing difficulties? What are the biggest impediments to a developer's productivity and how can we best help to increase a developer's productivity?

Traditionally, researchers aiming to increase developers' productivity, have focused on what developers have done, measuring their output and collecting data from software repositories. For instance, several approaches have been developed to automatically detect defects in the code based on metrics such as code churn or code complexity [Nagappan and Ball, 2005], [Zhang et al., 2007]. These approaches can help in decreasing maintenance cost and effort, however, they have two general downsides: first, the used metrics can only be calculated after change tasks are completed and second, they do not take into account the individual differences between developers, such as the experience level. With the Personal Software Process (PSP) Humphrey, as one of few, focused more on the individual by helping them to improve their skills and quality of work, but PSP requires developers to track measures manually, such as their schedule data, which is cumbersome and only allows more coarse granular insights [Humphrey, 1996].

Emerging biometric (*aka.* psychophysiological) sensor technology offers new ways to measure more of a developer, such as her cognitive and emotional states while she is working, rather than just the outputs of her work. The idea behind most biometric sensor technology is to measure physiological features of a person that can in turn be linked to the person's psychological states. As an example, a person who is stressed generally tends to sweat more than in less stressful situations and this difference in sweat leads to a varying electronic conductance of the skin that can be measured by electro-dermal activity (EDA) sensors. Extensive research in psychology has already investigated and correlated biometric measures, including skin-, heart-, eye- and brain-related ones, with a person's cognitive and emotional states. For instance, researchers have found that brain- and skin-related measurements can be linked to mental and cognitive load [Kramer, 1991], [Gevins et al., 1998], [Berka et al., 2007], [Nourbakhsh et al., 2012], [Setz et al., 2010].

Research in software engineering is also starting to take advantage of biometric data to better understand what developers are going through in their work, measure their productivity, and to overall improve their productivity and wellness. With the recent advances, biometric sensors are becoming increasingly

less invasive and are easier to accept and integrate into a developer's work without being bound to specific tasks, computers or locations. At the same time, the advances allow to capture more fine-grained biometric data in real-time, which offers new opportunities for more instantaneous support and feedback to developers. The vision is to integrate biometric sensing into a developer's work and use the data to ensure a developer's time is spent as productive as possible. In particular, biometrics might be used to measure aspects such as the flow and progress of individual developers or the difficulty they experience. These measures could then be used to provide instantaneous support, for instance, to avoid interruptions at inopportune moments, detect difficult parts in the code, and to intervene before a developer creates a bug.

In this paper, we will present an overview of the use of biometric sensors in the context of software development in general and more specific findings from our initial studies that demonstrate the potential that biometric data can have to accurately and instantaneously measure perceived task difficulty, progress and interruptibility of a developer. This offers much promise to provide better developer support and improve individual productivity. At the same time there are still several challenges to overcome for this to become a reality and widely accepted by developers, such as privacy concerns or sensor limitations that we will discuss as well.

The paper is structured as follows. First, we provide background information on biometric data and the measures commonly used in research (Section 6.2). Second, we present a general biometric sensing approach in software development (Section 6.3) and discuss findings of initial research in the area, including ours (Section 6.4). Then, we discuss future opportunities and challenges (Section 6.5) before we conclude (Section 6.6).

6.2 Biometric Data

Psychophysiology explores the relation between psychological states and processes and their physiological reactions [Andreassi, 2007]. An increasing amount of research has shown that specific cognitive and emotional states, such as high or low

cognitive load, or arousal and valence, can be correlated with biometric measures, such as electro-dermal activity or pupillary response [Wilson, 2002], [Richter et al., 1998], [Murugappan et al., 2008], [Haag et al., 2004], [Iqbal et al., 2004]. These psychological states and processes are influenced by the person and the task that is being performed, and in turn affect the outcome. For instance, according to the cognitive load theory [Sweller et al., 2011], cognitive load — the required mental effort to perform a task — is composed of intrinsic, extrinsic and germane load, including aspects such as the inherent task difficulty, the task format, as well as the person's age, experience and personality traits. The cognitive load experienced by a person during a task affects aspects, such as the likelihood of errors being created, a person's interruptibility and performance [Ko and Myers, 2005], [Iqbal and Bailey, 2005], [Ayres, 2001].

Similar to cognitive load, valence and arousal are concepts that can be influenced by various factors, such as task difficulty and personality traits [Basch and Fisher, 1998], [Plutchik and Conte, 1997], [Hutt and Weidner, 1993], and in turn can influence the outcome, such as the perceived progress and emotions [Brief and Weiss, 2002], [Graziotin et al., 2013], [Khan et al., 2011]. According to Russell's circumplex model, arousal and valence are the two cognitive dimensions of emotions [Russell, 1980], [Russell, 2003]. The arousal dimension indicates the amount of activation that is associated with an emotion, while the valence dimension is referring to the positive or negative character of the emotion. Various studies in the area of psychology have shown that biometrics can be used to determine the arousal and the valence dimension of emotions [Murugappan et al., 2008], [Murugappan et al., 2009], [Chanel et al., 2008].

Based on the link between biometrics and psychological states, biometrics might allow us to better understand what a developer experiences during work and to accurately predict outcome aspects, such as the error rate. An overview of these concepts in the software development context is given in Figure 6.1. Biometric measurements can roughly be divided into five different categories according to the origin of the measurements: eye-related, brain-related, skin-related, heart-related and breathing-related measurements. Table 6.1 provides an overview of some of these measures together with the psychological states

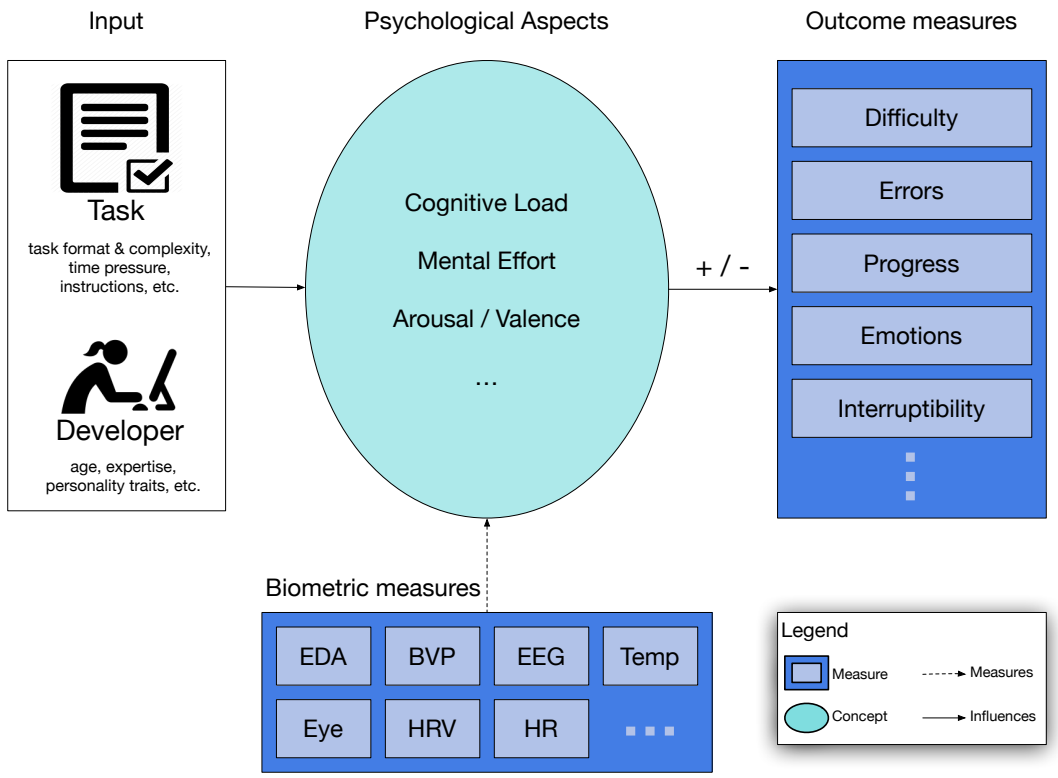


Figure 6.1: Exemplary illustration of psychophysiological relations in a software development context.

and processes that previous research, mostly in psychology, has already linked them to.

Eye-related Measurements. Interesting eye-related features are the eye movement, the pupil size and the eye blinks. Eye movement can further be separated into fixations, when the eye gaze fixates on a specific, non-moving object, and saccades that refer to the moving of the gaze point from one object to another. Most of these features can be captured with an eye tracker that uses the reflection of infrared light from the eyes, but eye blinks can also be extracted from EEG data [Manoilov, 2007]. The average eye blink rate lies between 15 and 20 blinks per minute, but can increase significantly when a person is tired, experiences a lot of stress or is under time pressure [Andreassi, 2007]. Previous research has linked eye-related features to cognitive, mental and memory load [Brookings

Table 6.1: Overview of some biometric measures and previously found links from literature.

Measure	Previously linked to
Eye-related	
Pupil size	cognitive & mental load [Haapalainen et al., 2010, Iqbal et al., 2004]; excitement [Muldner et al., 2010];
Fixations	cognitive load [Ikehara and Crosby, 2005]; valence [Carniglia et al., 2012]
Eye blinks	mental workload [Brookings et al., 1996]; frustration, stress, anxiety [Kapoor et al., 2007], [Doehring, 1957]
Brain-related	
Frequency bands (FBs)	mental workload [Brookings et al., 1996]; valence, arousal [Sammler et al., 2007], [Reuderink et al., 2013]; happiness and sadness [Li and Lu, 2009]
Ratios of FBs	task engagement [Kramer, 1991]; valence, arousal [Lin et al., 2010]
Skin-related	
Electro-dermal activity (EDA)	valence, arousal, engagement [Haag et al., 2004], [McDuff et al., 2012]; frustration [Freeman, 1940], [Kapoor et al., 2007]; stress and cognitive load [Setz et al., 2010]
Skin temperature	task difficulty [Anthony et al., 2011]; valence, arousal [Haag et al., 2004]; boredom, engagement, anxiety [Chanel et al., 2008];
Heart-related	
Heart rate (HR)	mental load & effort [Richter et al., 1998], [Veltman and Gaillard, 1998]; valence, arousal [Haag et al., 2004], [Sammler et al., 2007]; positive / negative affect [Drachen et al., 2010]; happiness [Steptoe et al., 2005]
Heart rate variability (HRV)	mental effort [Veltman and Gaillard, 1998]; task difficulty [Walter and Porges, 1976]; anxiety [Rani et al., 2004]; various emotional states [McCraty and Tomasino, 2006]
Blood volume pulse (BVP)	cognitive load [Peper et al., 2007]; various emotions [Picard et al., 2001]; valence, arousal [Haag et al., 2004];
Breathing-related	
Respiratory rate	mental effort [Veltman and Gaillard, 1998]; task difficulty [Kuznetsov et al., 2011]

et al., 1996], [Haapalainen et al., 2010] as well as emotional aspects, such as valence or negative affect [Carniglia et al., 2012], [Muldner et al., 2010].

Brain-related Measurements. The varying activity of neurons in the brain causes fluctuations in the voltage potential along the scalp that can be measured with an electroencephalogram (EEG) [Andreassi, 2007]. Research has identified a number of brain wave frequency bands from EEG data that are called alpha (α), beta (β), gamma (γ), delta (δ), and theta (θ). Each of these brain wave frequency bands has a specific frequency range and amplitude and exhibits more or less activity under certain circumstances. For instance, alpha waves can typically be observed when an individual is in a relaxed state, but the alpha waves either disappear or their amplitude decreases significantly as soon as the physical or mental activity increases [Andreassi, 2007]. Generally, research has linked these specific frequency bands and ratios thereof with mental workload, task engagement and emotions [Kramer, 1991], [Brookings et al., 1996], [Li and Lu, 2009].

Skin-related Measurements. Common skin-related measurements are the skin temperature and the electro-dermal activity (EDA), formerly also known as galvanic skin response (GSR). EDA measures the electrical conductance of the skin. As an example, when an individual is aroused, the sweat glands in the skin will produce more sweat and the electrical conductance of the skin will therefore increase. The EDA signal consists of two parts: the slowly changing, low frequency, tonic part, and the fast adapting, high frequency, phasic part [Schmidt and Walach, 2000]. Commonly used features for the tonic part and the temperature signal are the mean value or the area under the curve (AUC); commonly used features for the phasic part are related to the peaks in the signal. EDA as well as skin temperature have previously been correlated with the general arousal level and also with specific emotions [Ekman et al., 1983], [Haag et al., 2004], [Boucsein, 2012].

Heart-related Measurements. For heart-related measurements we focus on three different features: the heart rate (HR), the heart rate variability (HRV) and the blood volume pulse (BVP). The heart rate refers to the number of contractions of the heart each minute and the heart rate variability represents the

variation in the time interval between two consecutive heart beats. The blood volume pulse measures the blood flow through specific parts of the body and may change when the sympathetic nervous system increases its activity, for instance because of stress [Andreassi, 2007]. In research, HR and HRV have been linked to mental and cognitive load, as well as stress levels and emotional states [Richter et al., 1998], [Veltman and Gaillard, 1998], [McCraty and Tomasino, 2006]. BVP has predominantly been correlated with various emotions [Picard et al., 2001]. Common features of these measurements are the mean heart rate, the mean and the standard deviation of the time between two heart beats and features that capture the peaks of the BVP signal.

Breathing-related Measurements. We only used one breathing-related measurement, the respiratory rate (RR). The respiratory rate refers to the number of breaths within a specific time period and under normal conditions it is in the range between 12 and 15 breaths per minute [Cacioppo et al., 2007]. Research has used the respiratory rate to assess mental effort, task difficulty and task demand [Veltman and Gaillard, 1998], [Kuznetsov et al., 2011]. Commonly used features are the mean respiration rate or the $\log_{10}$ variance of the respiration signal.

6.3 Biometric Sensing in Software Development

Biometric data has the potential to provide insights on what developers are experiencing in their work in real-time, for instance, when they are getting into the flow and are highly focused or when they are having difficulties and are getting frustrated. One of the early approaches in the software development domain mentioning biometric sensor technology is the Ginger2 environment by Torii and colleagues that included an eye-tracker and a skin resistance level sensor to empirically study developers [Torii et al., 1999]. In the context of software development, biometric sensors have been and are being used predominantly to gain a better understanding of developers' program comprehension either using eye-tracking [Bednarik and Tukiainen, 2006], [Crosby and Stelovsky, 1990], [Rodeghero et al., 2014], [Kevic et al., 2015] or by measuring brain ac-

tivity [Ikutani and Uwano, 2014], [Siegmund et al., 2014]. Recently, especially with the advances in sensor technology and the availability of more affordable biometric sensors, a few software engineering researchers have also looked at other aspects, for instance, using measures of cerebral blood flow, measures of sub-vocal utterances captured with electromyography (EMG), or EDA, eye-tracking and brain activity measures to asses task difficulty of small code snippets or programming tasks [Nakagawa et al., 2014], [Parnin, 2011], [Fritz et al., 2014a]. In a few cases, biometric measures have also been used for studies with software developers on longer and more realistic development tasks or even in the field [Müller and Fritz, 2015b], [Züger and Fritz, 2015], [Müller and Fritz, 2016].

To study the potential of biometric data for any particular aspect in the software development domain, there are a few general questions and steps that researchers have to address, such as which sensors to use, how to setup the study and how to analyze the data. In the following, we will discuss these three points and provide insights from our own experiences on the use of biometric sensors, before we will provide more detailed information on the use of biometric data to measure developers' perceived task difficulty, progress and interruptibility in Section 6.4.

6.3.1 Which Biometric Sensors to Use?

There is already a plentitude of devices available for a broad audience that contain some sort of biometric sensors, such as the Mio Fuse[1], the Microsoft Band[2] or the Apple Watch[3] that can track a person's heart rate. At this point, however, most of these devices do not yet support the granularity, sampling rate or the specific biometric features that are required and that were previously linked to psychological states and processes, such as high or low cognitive load. Therefore, more specialized biometric sensors are still needed for these kinds of studies, and it is not always easy to find a good set. Some of the questions to consider before choosing a set of sensor devices are: which biometric features have been shown

[1] `http://www.mioglobal.com`
[2] `http://www.microsoft.com/microsoft-band`
[3] `http://www.apple.com/watch`

to capture phenomena similar to the one of interest (*e.g.*, in psychology research) and which set of sensors captures these; do the devices provide the necessary accuracy, granularity and sampling rate; do the sensor devices allow to conduct the research without obstructing/constraining the developer (too much); and is an API available to collect the biometric data from the devices as needed.

In our studies, we used seven different sensors so far: a Tobii TX300 eye tracker[4] or an Eyetribe eye tracker[5] for eye-related measures, an Affectiva[6] Q Sensor 2.0 (no longer available), or an Empatica E3 or E4 wristband[7] to record various skin- and heart-related measures, a Neurosky Mindband[8] to capture brain-related measures, and finally a SenseCore chest strap (also no longer available) to record various skin-, heart-, and breathing-related measurements. These specific sensors were chosen for three reasons: first, research in psychology has linked the features recorded with these sensors to outcome measures we are interested in, second, these sensors were minimally invasive for the measures they recorded, and third, these sensors were reasonably priced and affordable for a single developer, with the exception of the Tobii eye-tracker. The major challenges we encountered were due to a type of sensor ceasing to exist or being supported either because the company shifted focus or because the company actually closed down, and also due to the lack of mature APIs to collect certain data in real-time.

6.3.2 Setting up the Study

Most biometric features and sensors are sensitive to various variables, such as lighting and noise, the exact placing of the sensor devices, the time of day, or the weather [Cacioppo et al., 2007], to name just a few. Therefore, when setting up a study with biometric sensors or when employing them in the field, it is important to think about how to best setup the study to examine the phenomena of interest. Relevant questions to address are: how do you ensure that the study conditions are the same or almost the same for all participants; should the study

[4] `http://www.tobiipro.com/product-listing/tobii-pro-tx300/`
[5] `https://theeyetribe.com/`
[6] `http://www.affectiva.com/`
[7] `https://www.empatica.com`
[8] `http://neurosky.com/biosensors/eeg-sensor/`

take place in a lab to better control the environmental variables, or in the field to examine whether the approach could also be used in more realistic settings; is it possible to capture the biometric data for longer periods or only for very short tasks; and how to best collect the outcome measure as well as a baseline of each biometric feature to normalize the captured biometric data and account for individual differences in the biometric data.

Our recent studies ranged from controlled lab experiments in which developers worked on short and small predefined code tasks to multiple days field studies with professional developers working in their usual environment on their usual tasks. Each study was designed to be minimally invasive in terms of time and impact on a participant's usual workflow to avoid biasing the results. Figure 6.2 depicts a participant in our field study on interruptibility [Züger and Fritz, 2015] sitting in front of his work station while wearing an Empatica E3 wristband and a Neurosky Mindband. For this research, we ran a lab and a field study and found that biometric data can be used in both cases to predict interruptibility of developers, but that the predictions are more accurate in the lab. For the outcome measure, we used the Tablet to trigger interruptions and to collect user ratings on interruptibility and disruptiveness. In addition, we asked study participants to watch calming two-minute movies of fish swimming in a fish tank as well as to relax and not think of anything specific during that time period. We used the biometric data collected during the second minute of the movie as a baseline for each participant.

6.3.3 Data Analysis

Once the data is recorded, several steps to clean it and extract specific features of interest have to be performed. An overview of the general approach we followed in our studies to achieve this goal is presented in Figure 6.3.

Since biometric data is notoriously noisy, the first step is to clean the captured data. Depending on the kind of biometric data, different noise cleaning and filtering techniques have to be applied. For instance, for eye-tracking data the invalid data points that are labeled as such by the eye-tracker should be removed. For the EDA data an (exponential) smoothing filter can be applied to remove

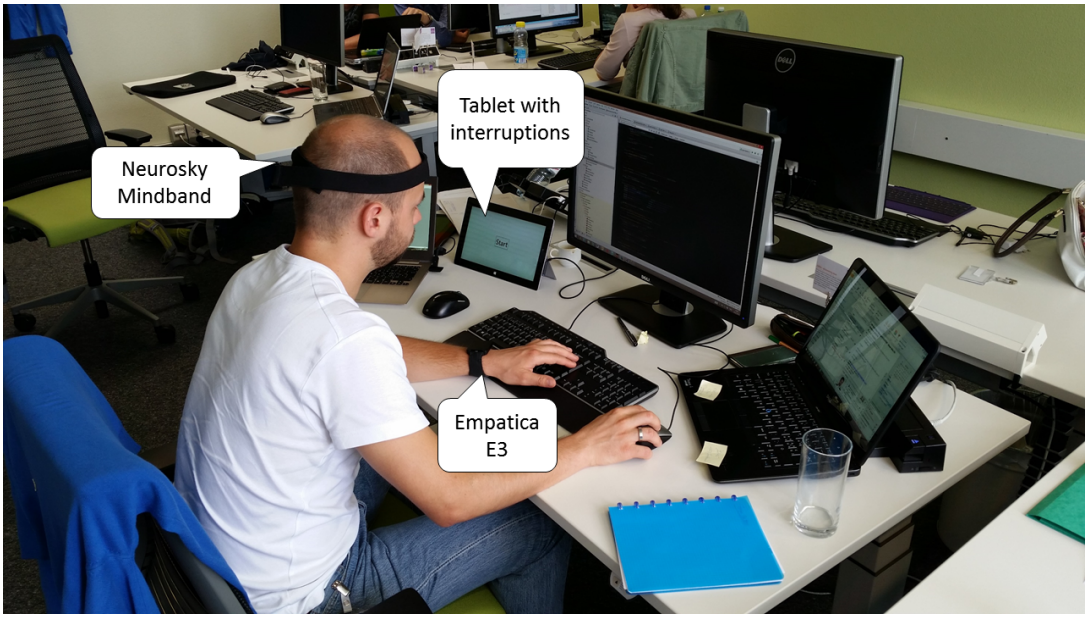


Figure 6.2: Field study setup with a participant wearing a headband and a wristband. The tablet was used to trigger interruptions.

noise and the EDA signal's DC component should be subtracted to base it at 0 µS. Most of these noise cleaning techniques are described in literature, but also require a careful analysis of the captured data. During this cleaning process, it is also advantageous to segment the collected data as needed to reduce the amount of data that has to be processed for later analysis steps. In our previous study on predicting a developer's emotions and progress for example, we only used and processed ten-second time windows of biometric data collected just before each time we interrupted developers and asked them to rate their emotions and progress. For the segmentation it is important to make sure that the segments are independent from each other with respect to later analysis steps. Especially since biometric features, such as body temperature or the tonic part of the EDA signal, take some time to get back to a baseline or original value, the segmentation has to take this into account.

In many cases, the raw (and cleaned) data by itself is not meaningful and specific features have to be extracted. For instance, commonly extracted features from HRV data are the mean and the standard deviation of the time between two

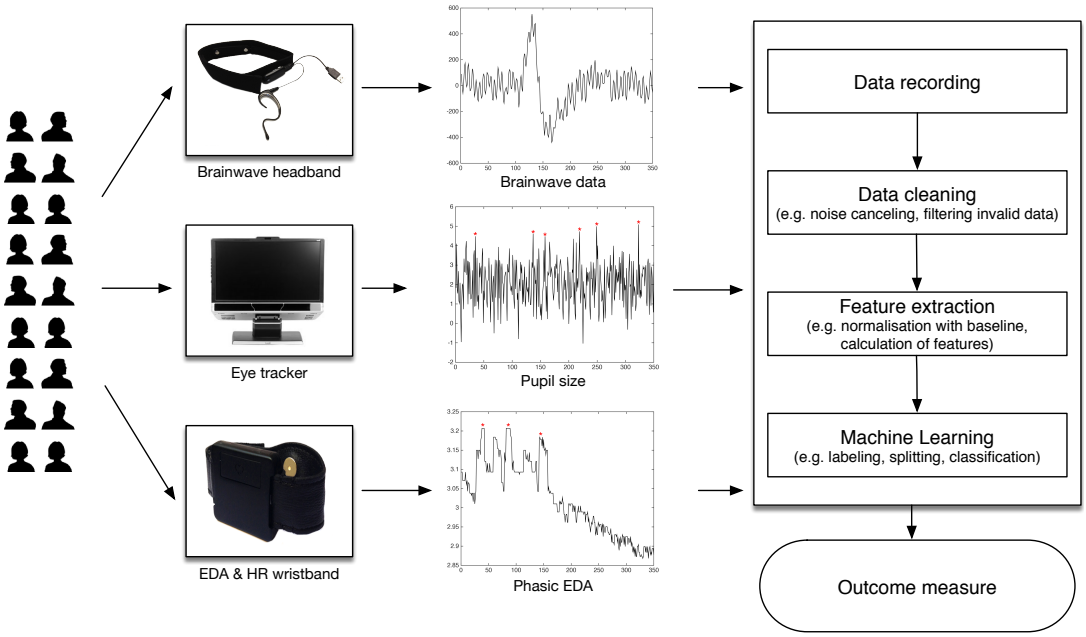


Figure 6.3: Overview of the general approach to record and analyze biometric data.

heart beats, and EEG data is commonly split up into five brain wave frequency bands ($\alpha - \theta$). Since biometric data is very individual, these features also have to be normalized by participant. In our previous studies, we used a biometric baseline collected during fish tank movies for the normalization.

Before feeding the extracted features into a machine learning classifier, the data needs to be labeled and split. For the data labeling, the outcome measure under analysis has to be assigned to the biometric data segments used for predicting the outcome measure. As an example from our research, Figure 6.4 presents biometric data — EDA and HR — that we collected during one of our studies in combination with the participant's valence and progress ratings that it was then labeled with [Müller and Fritz, 2015b]. Valence refers to the positive or negative character of an experienced emotion, the higher the rating the more positive the emotion. In the depicted data sample, a difference is visible between the EDA signal during the phases of medium progress and positive valence rating, and the phase in which the developer got stuck and perceived negative emotions.

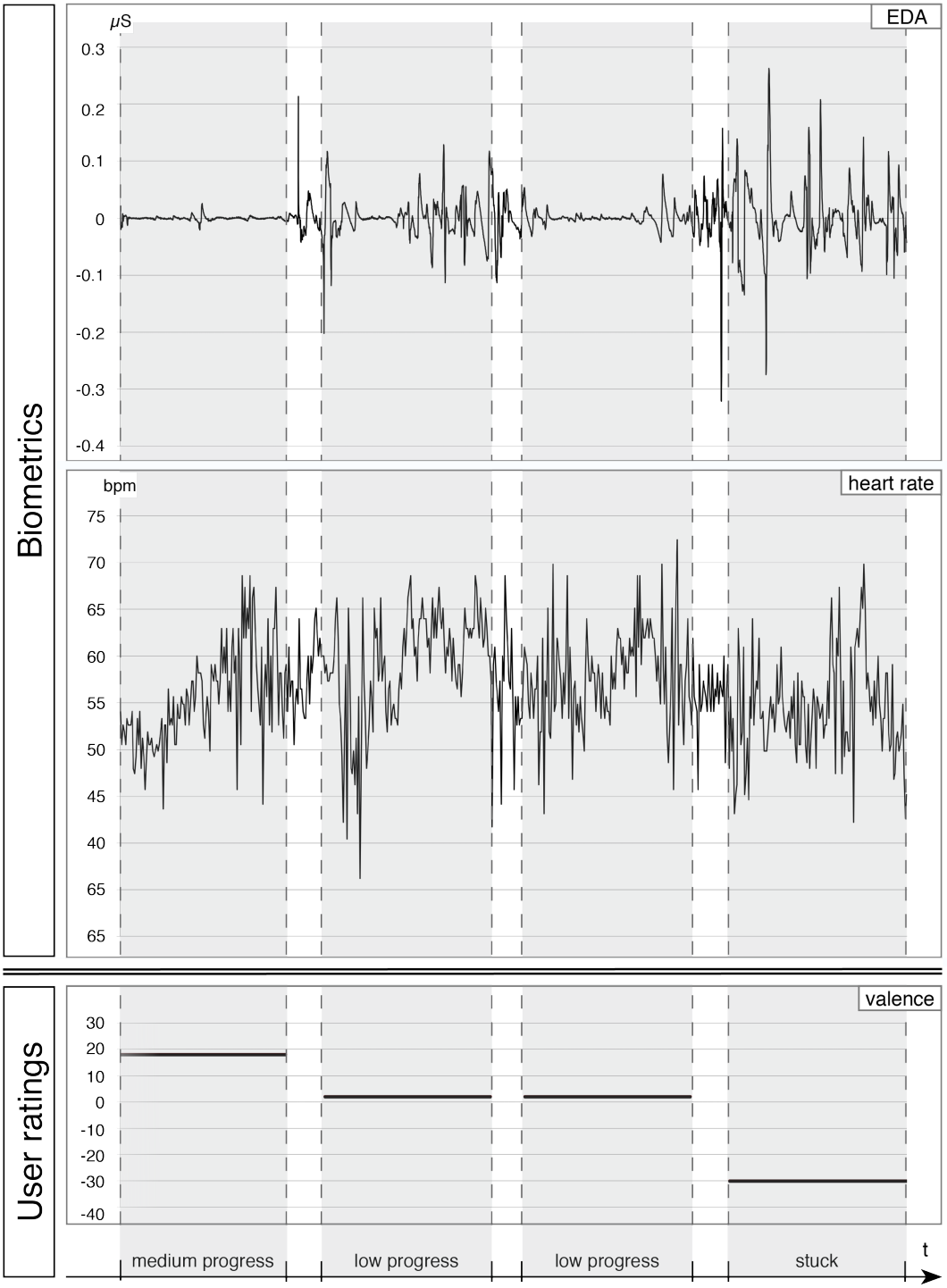


Figure 6.4: Biometric data and the participant's perceived progress and valence ratings collected during one of our studies.

In a next step, the data has to be split into training and test data. Depending on the evaluation method (*e.g.*, cross-validation or leave-one-out), different methods to perform the splitting have to be considered. In all cases, it is important to ensure the training and test data set do not overlap or the splitting biases the predictions in any way.

6.4 Sensing Difficulty, Progress and Interruptibility

One of the most common reasons for professional developers to have a productive workday is getting into the flow and making lots of progress without having many context-switches, interruptions or distractions [Meyer et al., 2014]. While it was previously difficult or sometimes impossible to measure aspects such as the developer's perceived progress or interruptibility in real-time, the advances in biometric sensors might provide us the means to measure them. In the following, we will discuss the value and feasibility of using biometric sensors in the software development domain by focusing on three such aspects that we also explored in our own research: task difficulty, progress and interruptibility.

6.4.1 Sensing Task Difficulty

Knowing when and for which code or tasks developers experience difficulties might allow us to lower development and evolution cost, for instance by identifying quality concerns in the code early on and by intervening before developers create bugs. Research on manually detecting code quality concerns has shown that code reviews can help significantly to discover and improve code with quality concerns, including defects [Bacchelli and Bird, 2013], [Bosu et al., 2015], [Rigby et al., 2008]. However, manual inspections require time and effort and can only be done after. Most research to automatically determine task and code difficulty as well as predicting defects has predominantly focused on the use of various code metrics, such as complexity metrics, size metrics, or code churn [Kasto and Whalley, 2013], [Katzmarski and Koschke, 2012], [Feigenspan et al., 2011], [Nagappan and Ball, 2005], [Gray et al., 2009]. These metrics, however, can mostly only

be computed after a developer finished a change task and also do not take into account the individual differences that exist between developers. A step towards more individual data of developers was taken by Lee and colleagues who defined micro interaction patterns for predicting defects [Lee et al., 2011].

Since research in psychology has shown that certain biometric features are linked to a person's mental effort for working on a task, biometric data has a great potential to help us assess the difficulty a developer is experiencing when working on a change task and when the developer might be close to creating an error in the code. Some preliminary work in software engineering has looked at the use of biometric measurements to determine task difficulty. For instance, Nakagawa *et al.* [Nakagawa et al., 2014] measured cerebral blood flow (CBF) to distinguish between two difficulty levels while developers were performing code comprehension tasks. Similarly, Parnin [Parnin, 2011] investigated the use of electromyography (EMG) to measure sub-vocal utterances and found that these measurements might be used to assess programming task difficulty.

In our own work with A. Begel, S. Yigit-Elliott and M. Züger, we conducted a controlled lab experiment with 15 professional software developers to examine the feasibility of using biometric sensors to assess the difficulty a developer experiences working on small code comprehension tasks [Fritz et al., 2014a]. Each participant was asked to perform eight short code comprehension tasks while sitting in front of a computer with an eye-tracker and wearing an EDA and an EEG sensor. For each task participants were asked to read a small C# code snippet and then answer a multiple choice question. To ensure varying levels of task difficulty, we altered tasks in several ways, such as the use of obfuscated variable names instead of mnemonic ones, randomly-ordered field assignments or the use of loops with various levels of complexity. With these tasks and variations we wanted to stress participants' cognitive abilities, such as the working memory, their math and logic skills and their ability in spatial relations. For assessing perceived task difficulty, we asked all participants to rank and rate the tasks according to their difficulty.

Findings. We trained a Naïve Bayes classifier and found that we are able to use the collected biometric data to predict the perceived task difficulty for a new developer that we did not train on with 65.0% precision and 64.6% recall using all three sensors. The precision and recall went up to 84.4% and 69.8% respectively, when we predicted for a new task and trained the classifier on other tasks for each developer. So while the classifiers can be used to predict difficulty even for people they were never trained on, they can be a lot better in predicting when they are trained on the individual they are used for, since biometric data varies a lot across people. Our analysis also showed that, while subsets of the biometric sensors also lead to good results, a combination of all three sensors, eye-tracking, EDA and EEG, performed best. Finally, an analysis on sliding time windows of the biometric data also showed that even short windows of only a couple of seconds can achieve high precision and recall, illustrating the potential of measuring task difficulty in real-time [Fritz et al., 2014a].

6.4.2 Sensing Progress and Emotions

Developers feel particularly productive in a workday when they get into the flow, making lots of progress [Meyer et al., 2014]. If we are able to measure a developer's progress and flow in real-time, we would, for example, know when a developer is stuck and she might need help or even just a break, or we could use the information on a developer being in the flow to indicate to coworkers not to interrupt and avoid costly interruptions. Similarly, knowing more about a developer's emotions might allow us to help them when they are frustrated and would benefit from a break or the help of a coworker.

Several empirical studies have investigated the kind of emotions developers experience, their change over time and their correlation with developers' productivity, also, for instance, by inducing moods and measuring the impact on developers' performance [Shaw, 2004], [Graziotin et al., 2013], [Graziotin et al., 2014], [Wrobel, 2013], [Khan et al., 2011]. Khan *et al.* also investigated whether keyboard and mouse input could be used to measure a developer's mood, but no generic correlation across all study participants was found [Khan et al., 2013]. Concerning the classification of progress, only very few studies have been

conducted in the field of software engineering. Carter *et al.*, for instance, mined IDE interaction logs to automatically determine when a developer is stuck and cannot make any progress [Carter and Dewan, 2010b].

In one of our studies, we examined the use of biometric data to assess a developer's progress and emotions, especially since research in psychology has already linked progress and emotions [Müller and Fritz, 2015b]. For this study, we had 17 participants (6 professional software developers, 11 Computer Science PhD students) work on two change tasks on open source systems for 30 minutes each in a quiet lab while again sitting in front of an eye-tracker and wearing a wristband to track EDA and heart-related measures and an EEG sensor. We used experience sampling, interrupting participants approximately every five minutes and asking them to rate their progress on a 5-point Likert scale and their emotions according to Russell's 2-dimensional Circumplex model on a valence and an arousal scale each from -200 to +200 [Russell, 1980]. As a baseline for our emotion-related measures, we also showed participants sets of pictures that are known to induce positive and negative emotions [Dan-Glauser and Scherer, 2011]. We used ten second time windows of biometric data just before each interruption and labeled these as either positive or negative emotion instances by using the baseline's ratings to split valence ratings during a change task as either positive or negative instances. For the progress ratings, we labeled the biometric data segments as high progress for ratings of 4 and 5 and low for 1 and 2, while we also removed neutral instances. We then used the collected and labeled 10 second time windows of biometric data to train and test a decision tree (J48) classifier using a leave-one-out method.

Findings. Using a decision tree classifier we were able to correctly classify cases as low or high progress in 67.7% of all cases (improving upon a naive classifier by 32.9% and a random one by 35.4%), and as negative or positive emotions in 71.4% of all cases (improving upon a naive classifier by 18.8% and a random one by 42.7%). The analysis also showed that there is a big variance between individuals and in how accurately one can predict progress and emotions using biometric data ranging from as little as 30.0% for one participant to getting all cases right

(100%) for another one. The biometric measures with the most predictive power for progress was thereby again a combination of all three biometric sensors: EDA tonic signal, skin temperature, brainwave frequency bands and pupil size. Overall, the results of this study provide further evidence on the value that biometric data can have to measure aspects of a developer during work and that it also works for longer and more realistic change tasks [Müller and Fritz, 2015b].

6.4.3 Sensing Interruptibility

Interruptions were one of the most commonly named reasons for decreasing developer productivity [Meyer et al., 2014]. Studies have actually shown that interruptions at inopportune moments will not only slow down a developer's work significantly, they will also lead to negative emotions and more errors in the code [Bailey and Konstan, 2006], [Czerwinski et al., 2004]. An automatic measure of interruptibility could thus significantly increase developers' productivity, for instance, by reducing in-person interruptions through visual cues to coworkers or by postponing computer-based interruptions, such as instant messages.

Much research on assessing interruptibility investigated the simulation and use of context-aware sensors, such as audio or video streams, keyboard or mouse input, active window information or information on task characteristics. For instance, Hudson *et al.* were able to classify interruptibility into two states with 78% accuracy by manually coding video and audio streams based on features, such as the phone being on the hook or people speaking [Hudson et al., 2003]. Fogarty *et al.* also simulated sensors — manually coding mouse and keyboard interactions — and were able to predict two states of interruptibility based on the interruption lag with 72% accuracy [Fogarty et al., 2005]. More recently, researchers started to explore the use of context-aware sensors with no need for manual coding. Tani *et al.* used pressure sensors to measure the force applied when typing on the keyboard and using the mouse. They were able to classify interruptibility into two states with an accuracy of around 70% [Tani and Yamada, 2013]. Ho *et al.* used accelerometers to identify physical activity transitions and found that interruptions at activity transitions are perceived better than those at random times [Ho and Intille, 2005].

Fewer research looked at the use of biometric sensors to assess interruptibility. Mathan *et al.* used EEG data to classify interruptibility during a US army urban combat training mission [Mathan et al., 2007]. Bailey *et al.* focused on eye-related measurements (pupil size) to infer the mental workload of study participants working on a goal-directed task. They found that a user's mental workload decreases at (sub)task boundaries suggesting that interruptions are best at these boundaries when the workload is low and fewer resources are needed to resume the task [Bailey and Iqbal, 2008]. Finally, Chen *et al.* used heart rate variability (HRV) and an electromyogram (EMG) and found a strong and significant correlation between the biometric measurements and the users' self-reported interruptibility during a variety of short and simple tasks [Chen et al., 2007].

In our work, we extended these studies by exploring the use of biometric sensors in real-world working contexts of software developers [Züger and Fritz, 2015]. In particular, we conducted two studies, a lab and a field study. In the lab study, we had eight graduate students work on three realistic change tasks on JHotDraw for a total of 60 minutes per student. For the field study, we recruited and visited ten professional software developers from four different companies and had them work on their own tasks in their real-world office environments for two hours each. Participants were told to work as usual without restriction on their activities. For both studies, participants were asked to wear a headband to record EEG and eye blink data and a wristband to record skin- and heart-related measures. During both studies, we triggered interruptions by playing a sound and changing the display on a tablet that we placed next to the developer's monitor(s) (see Figure 6.2). These interruptions were negotiated interruptions, *i.e.*, participants could decide for themselves when to address them, and were triggered at random time intervals that were between one and eleven minutes long. For each interruption we asked participants to perform a mental arithmetic exercise and to rate their interruptibility at the time of the notification and the perceived disturbance of the interruption each on a 5-point Likert scale. For our analysis we used time windows of biometric data ranging from ten seconds to three minutes ending with the triggered interruption. We labeled the time

windows with user ratings, once categorizing the 5-point scale into two states (interruptible or not) and once by keeping the fine-grained five state classification. For the machine learning we used Naïve Bayes since it outperformed Decision Trees and Support Vector Machine approaches with a ten-fold cross-validation.

Findings. For the lab as well as the field study we were able to use the biometric data to classify a developer's interruptibility with high accuracy into two states (lab: 91.5%, field: 78.6%) and into five states (lab: 43.9%, field: 32.5%). In all cases except for the five-state classification in the field, the trained classifier significantly outperforms a simple majority classifier. The fact that the lab study results are better than the field study ones and that the five-state classification for the field study did not significantly outperform the majority classifier, hints to the effect that external influences and more noise can have on such sensors in the field. The analysis also showed that shorter time windows of 10 seconds work generally better. Overall, these results illustrate the high potential of using biometric data as real-time indicators for interruptibility, and that they can be transferred to a real-world environment [Züger and Fritz, 2015].

6.5 Opportunities and Challenges

Overall, results of recent research studies demonstrate the high potential that biometric data has to measure certain aspects of a developer's work in real-time and with low-cost, off-the-shelf biometric sensors. In particular, the results show that biometric data, even captured in the field and only using short time windows, allows to accurately predict the difficulty developers experience working on small code comprehension tasks as well as their progress and interruptibility.

At some point in the future, we should be able to collect and leverage biometric data for each developer just the way that we are currently collecting and leveraging data on the artifacts that developers produce. However, instead of only calculating metrics on already completed change task data, we would then be able to know more about each individual developer while she is working on a change task. This will open up tremendous new opportunities for providing better

developer support and boosting the productivity of each individual developer. Especially with the fast advances and the pervasiveness of sensor technology, we might soon be able to collect valuable biometric data for a broad audience of software developers and take advantage of some of these opportunities in practice. At the same time, the sensitivity of the data and aspects of the technology also pose new challenges, such as privacy and ethical concerns.

6.5.1 Opportunities

Our research, in combination with results from other studies, provide initial evidence that biometric sensors can be used to determine various aspects of a developer's work, such as the perceived difficulty, progress and interruptibility. These findings open up new opportunities to better support every individual developer in real-time to boost their productivity and general well-being.

Measuring and Supporting Developers in Real-Time. Traditional approaches to assess task difficulty, code quality and a developer's productivity predominantly used various kind of code and process metrics defined on a developer's output. In most cases, these metrics are only available after a developer completed a task and thus only allow for a post-hoc analysis. Biometric data on the other hand has the potential for real-time measures and our study results have provided initial evidence on the accuracy and feasibility of such measures, especially given the short time windows of biometric data required.

Real-time measures on what a developer is experiencing, such as the difficulty of the task at hand or how focused the developer is at a given point in time also enable new possibilities for supporting developers while they are working. For example, a real-time measure of the difficulty a developer experiences while working on a specific code element might be used to automatically detect quality concerns in the code or when a developer is likely to create a bug in the code. This in turn would allow us to automatically prioritize code reviews and focus attention on the parts of the code that need it the most, and it might even enable us to prevent developers from creating bugs or committing them to a repository. Another example of the high potential of such biometric measures are real-time measures on a developer's interruptibility and flow. By knowing when a developer

is highly focused and in the flow, we are able to provide awareness to coworkers and thus decrease costly interruptions. At the same time, we could use such a measure in combination with persuasive technologies such as self-monitoring and goal-setting which could foster an increase in productivity, similar to the way that activity tracking devices help to increase and maintain physical activity over extended periods of time [Fritz et al., 2014b].

Measuring Individual Developers. Another drawback of traditional metrics is that they are mostly just based on artifacts and do not take into account the individual differences that exist between developers, such as their experience and expertise, but also their mood or general well-being that might influence their output and productivity. Biometric measures shift the focus to the individual developers. While this might require training such measures or classifiers for each individual developer, this will allow for better performance and more accurate predictions as our studies have shown.

Each person has times during the day or even days when they are more or less focused and productive. In psychology, some studies distinguish, for example, between morning and evening people. Measures that allow us to capture the cognitive and emotional states of each individual, will allow us to provide more tailored and valuable support. For instance, by knowing when a developer is most productive during the day or about their wellbeing, we might be able to optimize the work day and schedule the most demanding tasks for the times of the day she has the highest focus. We would also be able to provide more tailored and in-time recommendations, such as suggesting to talk to a coworker for help or taking a break when a developer is stuck and frustrated.

Boosting Productivity and Wellness. Overall, the insights that biometric data can provide us on a developer's work have immense potential to boost productivity. Preventing bugs, reducing quality concerns, avoiding interruptions at inopportune moments, providing a retrospective analysis, and automatically scheduling a productive day for developers are just some of the opportunities that the biometric data might enable. All of these can significantly improve a developer's work flow and productivity as well as reduce software evolution cost. Once we start taking advantage of biometric data, there are plenty of

opportunities to amplify the human smarts and ingenuity in the development process.

Biometric data might also be used to assess and foster the overall wellbeing of developers and in turn again their productivity. Several study results have already highlighted the correlation between progress and positive emotions, such as happiness. On a larger scale, it is also known that employees that are happy with their work and company have less sick days and are less likely to quit their job, which in turn increases the overall productivity of the company. With an increased awareness of the wellbeing of the developers on a team or in the company, managers might be able to react more quickly and provide a better work environment for their developers.

Pervasiveness of Smart Wearable Devices. The term smart wearable devices refers to electronic devices that are integrated in accessories or clothes and provide their users with some kind of real-time feedback about the user's activities or physiological features. The market for smart wearable devices is growing at an immense speed. From a market value of 600 million US$ in 2013, the market has increased to around 4 billion US$ in 2014 and will, according to forecasts, reach 30 billion US$ in 2020 [Hunn, 2014]. Wearable devices come in many different forms, such as smart glasses like the Google Glass[9], smart watches like the Apple Watch[10] or fitness tracker like the Fitbit[11].

The biometric sensors that are integrated in some of these devices are becoming more and more sophisticated and will soon support the capturing of biometric data that might be fine-grained and accurate enough for some of the presented scenarios. With the fast growing market and pervasiveness of smart wearable devices, an increasing amount of people will be wearing such devices. This will provide the opportunity to capture and collect biometric data for many developers on a daily basis and allow for the fine-tuning of classifiers on aspects such as task difficulty or progress which will further increase the potential the data has.

[9] `https://www.google.com/glass/start/`
[10] `http://www.apple.com/watch`
[11] `https://www.fitbit.com/`

6.5.2 Challenges

There are also several challenges with the use of biometric data, some of which we address in the following.

Sensor Limitations. To collect the relevant biometric data, you need sensors that are able to capture it and people that are willing to wear these sensors. Especially to collect data over longer periods of time, sensors need to be minimally invasive. In most cases though, there is a trade-off between invasiveness and the granularity and accuracy of the data that can be captured by a sensor. For instance, while some of our study participants agreed to wear a headband during a two hour study that allowed us to collect EEG data, most people would not wear this over several work days or even a whole work day and thus EEG data might not always be available.

While there are new and less invasive sensing devices coming out frequently, for instance the Apple Watch, it will still take some time until certain biometric features will be accessible over longer periods of time without disturbing the wearer. Furthermore, since several of these devices are still relatively new and immature, including their data transfer and the provided APIs, there are still quite a few challenges and obstacles to overcome to collect the relevant data. However, the fast growth of the market and technology should improve this situation soon.

Privacy & Ethical Concerns. The sensitivity and amount of the captured biometric data raises several privacy and ethical concerns. Using such data, for instance, to assess a developer's productivity or other work-related aspects might lead to a big brother effect and raise several red flags in developers. For our studies, we approached these concerns by ensuring to store the data in an anonymized way and by only making it accessible to the individual participant and the researchers. However, to provide the benefits to a broader audience of professional developers and possibly also their managers, future research has to look into means to ensure privacy and security of the biometric data. In particular, research has to examine which abstractions and aggregations of the data are most valuable while still ensuring enough privacy for the individual.

Recruiting Study Participants. Due to the sensitivity of the collected data, the invasiveness of the sensors, and general privacy and time concerns of software developers, finding participants for studies with biometric sensors is tedious and time-consuming. This situation is further aggravated by the fact that study participants can not always see an immediate benefit from participating. Especially for longer, several-day studies, it can be very challenging to find professional developers. This problem can be mitigated to some extent by choosing sensors that are as little invasive as possible, letting participants access their own biometric data and continuously motivating them to participate.

Noisy Data. Research has shown that biometric measures can be affected by the weather, the time of day, or personality traits [Cacioppo et al., 2007]. Thus the collected biometric data might contain a lot of noise and be affected by many other aspects than just the outcome measures, including individual differences. To minimize some of these effects, the environment in which a study will be conducted has to be chosen carefully, baselines should be established for each participant and normalization techniques should be applied. The results of our field studies provide initial evidence that biometric sensors can be used in noisy environments to make accurate predictions [Züger and Fritz, 2015], [Müller and Fritz, 2016]. With the new advances in sensor technology and also by training classifiers on bigger data sets of an individual, it might be possible to minimize the effects of the noise even further and provide more fine-grained classifications.

Choosing the Best Biometric Features. Research in psychology has analyzed various biometric features and found correlations to a lot of psychological aspects in some studies but none in others. This makes it difficult to know which features are best and which ones should be chosen for a machine learning classifier in a specific scenario. More research is therefore needed to establish which features are most promising in which contexts.

Analyzing Big Data. Typically, biometric sensors used for these kinds of studies have a relatively high sampling rate and generate big amounts of data even for smaller studies. For instance, over the course of our studies, we captured biometric data consisting of close to 140 million data points. This big amount of data leads to challenges in handling the data, cleaning the noise, normalizing

it and extracting the necessary features. Some of these issues can be tackled by having dedicated machines, and where possible, parallelizing the algorithms. By optimizing the data collection step *e.g.*, by focusing on a subset of features, reducing the sampling rate, or limiting the time windows, and with the advances in technology in general, it should be possible to further speed up the analysis process significantly in the near future.

6.6 Conclusion

Biometric data has the potential to reveal a lot about a developer's cognitive and emotional states in real-time. Initial results from several studies confirm this hypothesis and show that biometric data can actually be used to accurately predict certain aspects of a developer in real-time, such as the experienced task difficulty, the perceived progress and the developer's interruptibility. This offers much promise for improving developer support and boosting developer productivity overall, for instance, by automatically and instantaneously detecting code quality concerns or by reducing costly interruptions.

Our vision is to leverage biometric data on developers during their work just the way we are currently leveraging more traditional metrics, and then be able to provide developers with better and more individually tailored support. Especially with the fast advances in sensor and data analysis technology, we might soon all be wearing smart wearable devices with biometric sensors integrated that will already be accurate enough to provide some of this support. Given the sensitivity and the amount of biometric data collected per individual, there are however still several challenges to be addressed, in particular privacy concerns of the data and challenges for conducting research in the area.

Bibliography

[Ackerman et al., 1984] Ackerman, A. F., Fowler, P. J., and Ebenau, R. G. (1984). Software inspections and the industrial production of software. In *Proceedings of the Symposium on Software Validation*, pages 13 – 40.

[Affectiva, 2015] Affectiva (2015). Affectiva Q Sensor 2.0. `http://www.affectiva.com/`.

[Ali et al., 2012] Ali, N., Sharafi, Z., Guéhéneuc, Y.-G., and Antoniol, G. (2012). An empirical study on requirements traceability using eye-tracking. In *Proceedings of the 28th International Conference on Software Maintenance*, pages 191–200.

[Alikacem and Sahraoui, 2006] Alikacem, E. H. and Sahraoui, H. (2006). Generic metric extraction framework. In *Proceedings of the 16th International Workshop on Software Measurement and Metrik Kongress*, pages 383–390.

[Andreassi, 2007] Andreassi, J. L. (2007). *Psychophysiology: Human Behavior & Physiological Response*. Lawrence Erlbaum Associates.

[Andreessen, 2011] Andreessen, M. (2011). Why software is eating the world. The Wall Street Journal.

[Anthony et al., 2011] Anthony, L., Carrington, P., Chu, P., Kidd, C., Lai, J., and Sears, A. (2011). Gesture dynamics: Features sensitive to task difficulty and correlated with physiological sensors. *Stress*, 1418(360):312–316.

[AppleWatch, 2015] AppleWatch (2015). `http://www.apple.com/watch/`.

[Ax, 1953] Ax, A. F. (1953). The physiological differentiation between fear and anger in humans. *Psychosomatic Medicine*, pages 433–442.

[Ayres, 2001] Ayres, P. (2001). Systematic mathematical errors and cognitive load. In *Contemporary Educational Psychology*, pages 227–248.

[Bacchelli and Bird, 2013] Bacchelli, A. and Bird, C. (2013). Expectations, outcomes, and challenges of modern code review. In *Proceedings of the 35th International Conference on Software Engineering*, pages 712–721.

[Bailey and Iqbal, 2008] Bailey, B. P. and Iqbal, S. T. (2008). Understanding changes in mental workload during execution of goal-directed tasks and its application for interruption management. *ACM Transactions on Computer-Human Interaction*, 14(4):1–28.

[Bailey and Konstan, 2006] Bailey, B. P. and Konstan, J. A. (2006). On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state. *Computers in Human Behavior*, 22(4):685 – 708.

[Basch and Fisher, 1998] Basch, J. and Fisher, C. D. (1998). Affective events - emotions matrix: A classification of work events and associated emotions. *School of Business Discussion Papers*, 1998(65).

[Bauer et al., 1985] Bauer, L. O., Strock, B. D., Goldstein, R., Stern, J. A., and Walrath, L. C. (1985). Auditory discrimination and the eyeblink. *Psychophysiology*, 22(6):636–641.

[Beatty, 1982] Beatty, J. (1982). Task-evoked pupillary responses, processing load, and the structure of processing resources. *Psychological Bulletin*, 91(2):276–292.

[Bednarik and Tukiainen, 2006] Bednarik, R. and Tukiainen, M. (2006). An eye-tracking methodology for characterizing program comprehension processes. In *Proceedings of the Symposium on Eye Tracking Research & Applications*, pages 125–132.

[Bednarik and Tukiainen, 2008] Bednarik, R. and Tukiainen, M. (2008). Temporal eye-tracking data: evolution of debugging strategies with multiple representations. In *Proceedings of the Symposium on Eye Tracking Research & Applications*, pages 99–102.

[Bednarik et al., 2012] Bednarik, R., Vrzakova, H., and Hradis, M. (2012). What do you want to do next: a novel approach for intent prediction in gaze-based interaction. In *Proceedings of the Symposium on Eye Tracking Research & Applications*, pages 83–90.

[Berger, 1929] Berger, H. (1929). Über das Elektrenkephalogramm des Menschen. In *European Archives of Psychiatry and Clinical Neuroscience*, volume 87, pages 527–570.

[Berka et al., 2007] Berka, C., Levendowski, D. J., Lumicao, M. N., Yau, A., Davis, G., Zivkovic, V. T., Olmstead, R. E., Tremoulet, P. D., and Craven, P. L. (2007). EEG correlates of task engagement and mental workload in vigilance, learning, and memory tasks. *Aviation, Space, and Environmental Medicine*, pages B231–B244.

[Berntson et al., 1997] Berntson, G. G., Bigger, J. T. J., Eckberg, D. L., Grossman, P., Kaufmann, P. G., Malik, M., Nagaraja, H. N., Porges, S. W., Saul, J. P., Stone, P. H., and van der Molen, M. W. (1997). Heart rate variability: origins, methods, and interpretive caveats. *Psychophysiology*, 34(6):623–648.

[Bilchick and Berger, 2006] Bilchick, K. C. and Berger, R. D. (2006). Heart rate variability. *Journal of Cardiovascular Electrophysiology*, 17(6):691–694.

[Blatteis, 1998] Blatteis, C. M. (1998). *Physiology and Pathophysiology of Temperature Regulation*. World Scientific Publishing.

[Boehm et al., 2005] Boehm, B., Rombach, H. D., and Zelkowitz, M. V., editors (2005). *Foundations of Empirical Software Engineering: The Legacy of Victor R. Basili*. Springer.

[Boehm, 1981] Boehm, B. W. (1981). *Software engineering economics*. Prentice-Hall.

[Boehm et al., 1976] Boehm, B. W., Brown, J. R., and Lipow, M. (1976). Quantitative evaluation of software quality. In *Proceedings of 2nd International Conference on Software Engineering*, pages 592–605.

[Bosu et al., 2015] Bosu, A., Greiler, M., and Bird, C. (2015). Characteristics of useful code reviews: An empirical study at microsoft. In *Proceedings of the International Conference on Mining Software Repositories*, pages 146–156.

[Boucsein, 2012] Boucsein, W. (2012). *Electrodermal Activity*. Springer.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

[Brief and Weiss, 2002] Brief, A. P. and Weiss, H. M. (2002). Organizational behavior: affect in the workplace. *Annual Review of Psychology*, 53:279–307.

[Brookhuis and De Waard, 1993] Brookhuis, K. A. and De Waard, D. (1993). The use of psychophysiology to assess driver status. *Ergonomics*, 36(9):1099–1100.

[Brookings et al., 1996] Brookings, J. B., Wilson, G. F., and Swain, C. R. (1996). Psychophysiological responses to changes in workload during simulated air traffic control. *Biological Psychology: Psychophysiology of Workload*, 42(3):361 – 377.

[Burleson and Picard, 2004] Burleson, W. and Picard, R. (2004). Affective agents: Sustaining motivation to learn through failure and a state of stuck. In *Workshop on Social and Emotional Intelligence in Learning Environments*.

[Busjahn et al., 2015] Busjahn, T., Bednarik, R., Begel, A., Crosby, M., Paterson, J., Schulte, C., Sharif, B., and Tamm, S. (2015). Eye movements in code reading: Relaxing the linear order. In *International Conference on Program Comprehension*, pages 255 – 265.

[Butterworth, 1930] Butterworth, S. (1930). On the theory of filter amplifiers. *Wireless Engineer*, 7:536–541.

[Cacioppo et al., 2007] Cacioppo, J., Tassinary, L. G., and Berntson, G. G. (2007). *The Handbook of Psychophysiology*. Cambridge.

[Carniglia et al., 2012] Carniglia, E., Caputi, M., Manfredi, V., Zambarbieri, D., and Pessa, E. (2012). The influence of emotional picture thematic content on exploratory eye movements. *Journal of Eye Movement Research*, 5(4):1–9.

[Carter and Dewan, 2010a] Carter, J. and Dewan, P. (2010a). Are you having difficulty? In *Proceedings of the Conference on Computer Supported Cooperative Work*, pages 211–214.

[Carter and Dewan, 2010b] Carter, J. and Dewan, P. (2010b). Design, implementation, and evaluation of an approach for determining when programmers are having difficulty. In *Proceedings of the Conference on Supporting Group Work*, pages 215–224.

[Chanel et al., 2008] Chanel, G., Rebetez, C., Bétrancourt, M., and Pun, T. (2008). Boredom, engagement and anxiety as indicators for adaptation to difficulty in games. In *Proceedings of the 12th International Conference on Entertainment and Media in the Ubiquitous Era*, pages 13–17.

[Chen et al., 2007] Chen, D., Hart, J., and Vertegaal, R. (2007). Towards a physiological model of user interruptability. In *Human-Computer Interaction – INTERACT 2007*, volume 4663, pages 439–451.

[Cohen, 1960] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Education and Psychological Measurement*, 20:37–46.

[Collet et al., 1997] Collet, C., Vernet-Maury, E., Delhomme, G., and Dittmar, A. (1997). Autonomic nervous system response patterns specificity to basic emotions. *Journal of the Autonomic Nervous System*, 62(1–2):45 – 57.

[Colombetti, 2005] Colombetti, G. (2005). Appraising valence. *Journal of Consciousness Studies*, 12(8-10):103–126.

[Connor, 2011] Connor, A. M. (2011). Mining software metrics for the jazz repository. *Journal of Systems and Software*, 1(5):194–204.

[Cornforth et al., 2015] Cornforth, D. J., Koenig, A., Riener, R., August, K., Khandoker, A. H., Karmakar, C., Palaniswami, M., and Jelinek, H. F. (2015). The role of serious games in robot exoskeleton-assisted rehabilitation of stroke patients. In *Serious Games Analytics: Methodologies for Performance Measurement, Assessment, and Improvement*. Springer International Publisher.

[Crosby and Stelovsky, 1990] Crosby, M. and Stelovsky, J. (1990). How do we read algorithms? A case study. *Computer*, 23(1):25–35.

[Cunningham, 1993] Cunningham, W. (1993). The wycash portfolio management system. *ACM SIGPLAN OOPS Messenger*, 4(2):29–30.

[Curtis et al., 1979] Curtis, B., Sheppard, S., Milliman, P., Borst, M., and Love, T. (1979). Measuring the psychological complexity of software maintenance tasks with the Halstead and McCabe metrics. *Transactions on Software Engineering*, SE-5(2):96–104.

[Czerwinski et al., 2004] Czerwinski, M., Horvitz, E., and Wilhite, S. (2004). A diary study of task switching and interruptions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 175–182.

[Dan-Glauser and Scherer, 2011] Dan-Glauser, E. S. and Scherer, K. R. (2011). The geneva affective picture database (GAPED): a new 730-picture database focusing on valence and normative significance. *Behavior Research Methods*, 43(2):468–477.

[Dawson et al., 2007] Dawson, M., Schell, A., Filion, D., and Berntson, G. (2007). *Handbook of Psychophysiology*, chapter The Electrodermal System. Cambridge University Press.

[De Jong and Merckelbach, 1990] De Jong, P. J. and Merckelbach, H. (1990). Eyeblink frequency, rehearsal activity, and sympathetic arousal. *International Journal of Neuroscience*, 51(1-2):89–94.

[Détienne, 2002] Détienne, F. (2002). *Software Design - Cognitive Aspects*. Springer-Verlag.

[Dinarello and Porat, 2012] Dinarello, C. A. and Porat, R. (2012). *Harrison's Principles of Internal Medicine*, chapter Fever and Hyperthermia. McGraw-Hill.

[Doehring, 1957] Doehring, D. G. (1957). The relation between manifest anxiety and rate of eyeblink in a stress situation. Technical report, Central Institute for the Deaf, St Louis, MO.

[Drachen et al., 2010] Drachen, A., Nacke, L. E., Yannakakis, G., and Pedersen, A. L. (2010). Correlation between heart rate, electrodermal activity and player experience in first-person shooter games. In *Proceedings of the 5th Symposium on Video Games*, pages 49–54.

[Ebenau and Strauss, 1994] Ebenau, R. G. and Strauss, S. H. (1994). *Software Inspection Process*. McGraw-Hill, Inc.

[Eivazi and Bednarik, 2011] Eivazi, S. and Bednarik, R. (2011). Predicting problem-solving behavior and performance levels from visual attention data. In *Proceedings of the 2nd Workshop on Eye Gaze in Intelligent Human Machine Interaction*, pages 9–16.

[Ekkekakis, 2012] Ekkekakis, P. (2012). *Measurement in sport and exercise psychology*, chapter Affect, Mood, and Emotion. Human Kinetics.

[Ekman, 1994] Ekman, P. (1994). *The nature of emotion*, chapter Moods, emotions, and traits. Oxford Univserity Press.

[Ekman et al., 1983] Ekman, P., Levenson, R. W., and Friesen, W. V. (1983). Autonomic nervous system activity distinguishes among emotions. *Science*, 221(4616):1208–1210.

[Elish and Elish, 2008] Elish, K. O. and Elish, M. O. (2008). Predicting defect-prone software modules using support vector machines. *Journal of Systems and Software*, 81(5):649–660.

[Empatica, 2015] Empatica (2015). http://www.empatica.com.

[Ephrath, 1988] Ephrath, A. R. (1988). Breaking the software logjam: can we afford disposable software? In *Proceedings of the International Conference on Systems, Man, and Cybernetics*, pages 193–197.

[European Commision, 2013] European Commision (2013). European commission launches grand coalition for digital jobs. Press release.

[Eyetribe, 2015] Eyetribe (2015). Eyetribe. https://theeyetribe.com/.

[Fairclough et al., 2005] Fairclough, S. H., Venables, L., and Tattersall, A. (2005). The influence of task demand and learning on the psychophysiological response. *International Journal of Psychophysiology*, 56(2):171–184.

[Feigenspan et al., 2011] Feigenspan, J., Apel, S., Liebig, J., and Kastner, C. (2011). Exploring software measures to assess program comprehension. In *International Symposium on Empirical Software Engineering and Measurement*, pages 127 – 136.

[FindBugs, 2015] FindBugs (2015). http://findbugs.sourceforge.net/.

[Fitbit, 2015] Fitbit (2015). https://www.fitbit.com/.

[Fogarty et al., 2005] Fogarty, J., Ko, A. J., Aung, H. H., Golden, E., Tang, K. P., and Hudson, S. E. (2005). Examining task engagement in sensor-based statistical models of human interruptibility. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 331–340.

[Fowles et al., 1981] Fowles, D. C., Christie, M. J., Edelberg, R., Grings, W. W., Lykken, D. T., and Venables, P. H. (1981). Publication recommendations for electrodermal measurements. *Psychophysiology*, 18(3):232–239.

[Freeman, 1940] Freeman, G. L. (1940). A method of inducing frustration in human subjects and its influence upon palmar skin resistance. *The American Journal of Psychology*, 53(1):pp. 117–120.

[Fritz et al., 2014a] Fritz, T., Begel, A., Müller, S. C., Yigit-Elliott, S., and Züger, M. (2014a). Using psycho-physiological measures to assess task difficulty in software development. In *Proceedings of the 36th International Conference on Software Engineering*, pages 402–413.

[Fritz et al., 2014b] Fritz, T., Huang, E. M., Murphy, G. C., and Zimmermann, T. (2014b). Persuasive technology in the real world: A study of long-term use of activity sensing devices for fitness. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 487–496.

[Gevins et al., 1998] Gevins, A., Smith, M. E., Leong, H., McEvoy, L., Whitfield, S., Du, R., and Rush, G. (1998). Monitoring working memory load during computer-based tasks with EEG pattern recognition methods. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 40(1):79–91.

[Giger et al., 2012] Giger, E., D'Ambros, M., Pinzger, M., and Gall, H. C. (2012). Method-level bug prediction. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, pages 171–180.

[Goldberg and Kotval, 1999] Goldberg, J. H. and Kotval, X. P. (1999). Computer interface evaluation using eye movements: methods and constructs. *International Journal of Industrial Ergonomics*, 24(6):631 – 645.

[GoogleGlass, 2015] GoogleGlass (2015). `https://www.google.com/glass/start/`.

[Grady and Slack, 1994] Grady, R. and Slack, T. (1994). Key lessons in achieving widespread inspection use. *Software*, 11(4):46–57.

[Gray et al., 2009] Gray, D., Bowes, D., Davey, N., Sun, Y., and Christianson, B. (2009). Using the support vector machine as a classification method for software defect prediction with static code metrics. In Palmer-Brown, D., Draganova, C., Pimenidis, E., and Mouratidis, H., editors, *Engineering Applications of Neural Networks*. Springer Berlin Heidelberg.

[Graziotin et al., 2013] Graziotin, D., Wang, X., and Abrahamsson, P. (2013). Are happy developers more productive? the correlation of affective states of software developers ane their-self-assessed productivity. In *Proceedings of the 14th International Conference on Product-Focused Software Process Improvement*, pages 50–64.

[Graziotin et al., 2014] Graziotin, D., Wang, X., and Abrahamsson, P. (2014). Happy software developers solve problems better: psychological measurements in empirical software engineering. *PeerJ*, 2:e289.

[Greco et al., 2012] Greco, A., Lanata, A., Valenza, G., Rota, G., Vanello, N., and Scilingo, E. (2012). On the deconvolution analysis of electrodermal activity in bipolar patients. In *Proceedings of the Annual International Conference of the IEEE on Engineering in Medicine and Biology Society*, pages 6691–6694.

[Grimes et al., 2008] Grimes, D., Tan, D. S., Hudson, S. E., Shenoy, P., and Rao, R. P. (2008). Feasibility and pragmatics of classifying working memory load with an electroencephalograph. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 835–844.

[Gueorguieva and Krystal, 2004] Gueorguieva, R. and Krystal, J. H. (2004). Move over anova: Progress in analyzing repeated-measures data andits reflection in papers published in the archives of general psychiatry. *Archives of General Psychiatry*, 61(3):310–317.

[Haag et al., 2004] Haag, A., Goronzy, S., Schaich, P., and Williams, J. (2004). Emotion recognition using bio-sensors: First steps towards an automatic system. *Affective Dialogue Systems Lecture Notes in Computer Science*, 3068:36–48.

[Haapalainen et al., 2010] Haapalainen, E., Kim, S., Forlizzi, J. F., and Dey, A. K. (2010). Psycho-physiological measures for assessing cognitive load. In *Proceedings of the 12th International Conference on Ubiquitous Computing*, pages 301–310.

[Haarmann et al., 2009] Haarmann, A., Boucsein, W., and Schaefer, F. (2009). Combining electrodermal responses and cardiovascular measures for probing adaptive automation during simulated flight. *Applied Ergonomics*, 40(6):1026–1040.

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

[Handy, 2005] Handy, T. C. (2005). *Event-related Potentials: a Methods Handbook*. MIT Press.

[Hart and Staveland, 1988] Hart, S. G. and Staveland, L. E. (1988). Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Human mental workload*, 1(3):139–183.

[Hazlett, 2003] Hazlett, R. (2003). Measurement of user frustration: A biologic approach. In *Extended Abstracts on Human Factors in Computing Systems*, pages 734–735.

[Ho and Intille, 2005] Ho, J. and Intille, S. S. (2005). Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 909–918.

[Holland and Tarlow, 1972] Holland, M. K. and Tarlow, G. (1972). Blinking and mental load. *Psychological Reports*, 31(1):119–127.

[Holland and Tarlow, 1975] Holland, M. K. and Tarlow, G. (1975). Blinking and thinking. *Perceptual and Motor Skills*, 41(2):403–406.

[Hudson et al., 2003] Hudson, S., Fogarty, J., Atkeson, C., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J., and Yang, J. (2003). Predicting human interruptibility with sensors: A wizard of oz feasibility study. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 257–264.

[Humphrey, 1996] Humphrey, W. S. (1996). *Introduction to the Personal Software Process*. Addison-Wesley Professional.

[Hunn, 2014] Hunn, N. (2014). The market for smart wearables. WiFore Wireless Consulting.

[Hutt and Weidner, 1993] Hutt, J. and Weidner, G. (1993). The effects of task demand and decision latitude on cardiovascular reactivity to stress. *Behavioral Medicine*, 18(4):181–188.

[IDC, 2014] IDC (2014). Idc - analyze the future. `www.idc.com/getdoc.jsp?containerId=244709`.

[Ikehara and Crosby, 2005] Ikehara, C. S. and Crosby, M. E. (2005). Assessing cognitive load with physiological sensors. In *Proceedings of the 38th Hawaii International Conference on System Sciences*, page 295a.

[Ikutani and Uwano, 2014] Ikutani, Y. and Uwano, H. (2014). Brain activity measurement during program comprehension with NIRS. In *Proceedings of the International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 1–6.

[Iqbal and Bailey, 2005] Iqbal, S. T. and Bailey, B. P. (2005). Investigating the effectiveness of mental workload as a predictor of opportune moments for interruption. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, pages 1489–1492.

[Iqbal et al., 2004] Iqbal, S. T., Zheng, X. S., and Bailey, B. P. (2004). Task-evoked pupillary response to mental workload in human-computer interaction. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, pages 1477–1480.

[Jacob and Karn, 2003] Jacob, R. J. and Karn, K. S. (2003). Commentary on section 4 - eye tracking in human-computer interaction and usability research: Ready to deliver the promises. In *The Mind's Eye*, pages 573 – 605. North-Holland, Amsterdam.

[JHotDraw, 2014] JHotDraw (2014). http://www.jhotdraw.org/.

[Kapoor et al., 2007] Kapoor, A., Burleson, W., and Picard, R. W. (2007). Automatic prediction of frustration. *International Journal of Human-Computer Studies*, 65(8):724–736.

[Karahasanović et al., 2007] Karahasanović, A., Levine, A. K., and Thomas, R. (2007). Comprehension strategies and difficulties in maintaining object-oriented systems: An explorative study. *Journal of Systems and Software*, 80(9):1541–1559.

[Kasto and Whalley, 2013] Kasto, N. and Whalley, J. (2013). Measuring the difficulty of code comprehension tasks using software metrics. In *Proceedings of the Australasian Computing Education Conference*, pages 59–65.

[Katzmarski and Koschke, 2012] Katzmarski, B. and Koschke, R. (2012). Program complexity metrics and programmer opinions. In *Proceedings of the International Conference on Program Comprehension*, pages 17–26.

[Kennedy, 2015] Kennedy, N. (2015). Google mondrian: web-based code review and storage. http://www.niallkennedy.com/blog/2006/11/google-mondrian.html.

[Kevic et al., 2015] Kevic, K., Walters, B. M., Shaffer, T. R., Sharif, B., Shepherd, D. C., and Fritz, T. (2015). Tracing software developers' eyes and interactions for change tasks. In *Proceedings of the 10th Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering*, pages 202–213.

[Khan et al., 2013] Khan, I. A., Brinkman, W.-P., and Hierons, R. (2013). Towards estimating computer users' mood from interaction behaviour with keyboard and mouse. *Frontiers of Computer Science*, 7(6):943–954.

[Khan et al., 2011] Khan, I. A., Brinkman, W.-P., and Hierons, R. M. (2011). Do moods affect programmers' debug performance? *Cognition, Technology & Work*, 13(4):245–258.

[Khan et al., 2007] Khan, I. A., Hierons, R. M., and Brinkman, W. P. (2007). Mood independent programming. In *Proceedings of the 14th European Conference on Cognitive Ergonomics: Invent! Explore!*, pages 269–272.

[Klingner, 2010] Klingner, J. (2010). Fixation-aligned pupillary response averaging. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, pages 275–282.

[Ko and Myers, 2005] Ko, A. J. and Myers, B. A. (2005). A framework and methodology for studying the causes of software errors in programming systems. *Journal of Visual Languages & Computing*, 16(1):41–84.

[Ko et al., 2004] Ko, A. J., Myers, B. A., and Aung, H. H. (2004). Six learning barriers in end-user programming systems. In *Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing*, pages 199–206.

[Kramer, 1991] Kramer, A. F. (1991). Physiological metrics of mental workload: A review of recent progress. *Multiple-task Performance*, pages 279–328.

[Kreibig et al., 2007] Kreibig, S. D., Wilhelm, F. H., Roth, W. T., and Gross, J. J. (2007). Cardiovascular, electrodermal, and respiratory response patterns to fear- and sadness-inducing films. *Psychophysiology*, 44(5):787–806.

[Kuznetsov et al., 2011] Kuznetsov, N. A., Shockley, K. D., Richardson, M. J., and Riley, M. A. (2011). Effect of precision aiming on respiration and postural-respiratory synergy. *Neuroscience letters*, 502(1):13–17.

[Landis and Koch, 1977] Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

[Lanza and Marinescu, 2006] Lanza, M. and Marinescu, R. (2006). *Object-oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems*. Springer.

[Lavie et al., 2004] Lavie, N., Hirst, A., de Fockert, J. W., and Viding, E. (2004). Load theory of selective attention and cognitive control. *Journal of experimental psychology. General.*, 133(3):339–354.

[Lawrence J. Prinzel et al., 2001] Lawrence J. Prinzel, I., Alan T., P., Frederick G., F., Mark W., S., and Peter J., M. (2001). Empirical analysis of EEG and ERPs for pyschophysiological adaptive task allocation. Technical report, NASA Langley.

[Lawson, 1965] Lawson, R. (1965). *Frustration: The Development of a Scientitifc Concept*. The Macmillan Company.

[Lee and Tan, 2006] Lee, J. C. and Tan, D. S. (2006). Using a low-cost electroencephalograph for task classification in HCI research. In *Proceedings of the 19th Symposium on User Interface Software and Technology*, pages 81–90.

[Lee et al., 2011] Lee, T., Nam, J., Han, D., Kim, S., and In, H. P. (2011). Micro interaction metrics for defect prediction. In *Proceedings of the 19th Symposium and the 13th European Conference on Foundations of Software Engineering*, pages 311–321.

[Lehman, 1980] Lehman, M. M. (1980). On understanding laws, evolution, and conservation in the large-program life cycle. *Journal of Systems and Software*, 1:213–221.

[Leite et al., 2013] Leite, I., Henriques, R., Martinho, C., and Paiva, A. (2013). Sensors in the wild: Exploring electrodermal activity in child-robot interaction. In *Proceedings of the 8th International Conference on Human-Robot Interaction*, pages 41–48.

[Leppink and van den Heuvel, 2015] Leppink, J. and van den Heuvel, A. (2015). The evolution of cognitive load theory and its application to medical education. *Perspectives on Medical Education*, 4(3):119–127.

[Lessmann et al., 2008] Lessmann, S., Baesens, B., Mues, C., and Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *Transactions on Software Engineering*, 34(4):485–496.

[Levandovski et al., 2013] Levandovski, R., Sasso, E., and Hidalgo, M. P. (2013). Chronotype: a review of the advances, limits and applicability of the main instruments used in the literature to assess human phenotype. *Trends in Psychiatry and Psychotherapy*, 35:3 – 11.

[Levenson et al., 1990] Levenson, R., Ekman, P., and Friesen, W. (1990). Voluntary facial action generates emotion specific autonomic nervous system activity. *Psychophysiology*, 27:363–384.

[Li and Lu, 2009] Li, M. and Lu, B.-L. (2009). Emotion classification based on gamma-band EEG. *Conference Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1323–1326.

[Lin et al., 2010] Lin, Y.-P., Wang, C.-H., Jung, T.-P., Wu, T.-L., Jeng, S.-K., Duann, J.-R., and Chen, J.-H. (2010). Eeg-based emotion recognition in music listening. *IEEE Transactions on Biomedical Engineering*, 57(7):1798–1806.

[Liu and Setiono, 1996] Liu, H. and Setiono, R. (1996). A probabilistic approach to feature selection - a filter solution. In *Proceedings of the 13th International Conference on Machine Learning*, pages 319–327.

[Maaoui and Pruski, 2010] Maaoui, C. and Pruski, A. (2010). Emotion recognition through physiological signals for human-machine communication. *Cutting Edge Robotics*.

[Maimon and Rokach, 2006] Maimon, O. and Rokach, L. (2006). *Data Mining and Knowledge Discovery Handbook*, chapter Decision Trees. Springer.

[Makeig and Jung, 1996] Makeig, S. and Jung, T.-P. (1996). Tonic, phasic, and transient EEG correlates of auditory awareness in drowsiness. *Cognitive Brain Research*, 4(1):15 – 25.

[Mandryk, 2008] Mandryk, R. (2008). *Game Usability*, chapter Physiological Measures for Game Evaluation. CRC Press.

[Mandryk et al., 2006] Mandryk, R., Inkpen, K., and Calvert, T. (2006). Using psychophysiological techniques to measure user experience with entertainment technologies. *Special Issue on User Experience, Behaviour and Information Technology*, 25(2):141–158.

[Manoilov, 2007] Manoilov, P. (2007). Eye-blinking artefacts analysis. In *Proceedings of the International Conference on Computer Systems and Technologies*, page 52.

[Marinescu, 2004] Marinescu, R. (2004). Detection strategies: Metrics-based rules for detecting design flaws. In *Proceedings of the International Conference on Software Maintenance*, pages 350 – 359.

[Mark et al., 2014] Mark, G., Iqbal, S. T., Czerwinski, M., and Johns, P. (2014). Bored mondays and focused afternoons: The rhythm of attention and online activity in the workplace. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3025–3034.

[Martin and Turner, 1986] Martin, P. Y. and Turner, B. A. (1986). Grounded theory and organizational research. *The Journal of Applied Behavioral Science*, 22(2):141–157.

[Mathan et al., 2007] Mathan, S., Whitlow, S., Dorneich, M., Ververs, P., and Davis, G. (2007). Neurophysiological estimation of interruptibility: Demonstrating feasibility in a field context. In *Proceedings of the 4th International Conference of the Augmented Cognition Society*, page N/A.

[McConnell, 2004] McConnell, S. (2004). *Code Complete*. Pearson.

[McCraty and Tomasino, 2006] McCraty, R. and Tomasino, D. (2006). *Stress in Health and Deseases*, chapter Emotional Stress, Positive Emotions, and Psychophysiological Coherence. Wiley-VCH.

[McDuff et al., 2012] McDuff, D., Karlson, A., Kapoor, A., Roseway, A., and Czerwinski, M. (2012). Affectaura: an intelligent system for emotional memory.

In *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems*, pages 849–858.

[Meyer et al., 2014] Meyer, A. N., Fritz, T., Murphy, G. C., and Zimmermann, T. (2014). Software developers' perceptions of productivity. In *Proceedings of the International Symposium on the Foundations of Software Engineering*, pages 19–29.

[MicrosoftBand, 2015] MicrosoftBand (2015). `https://www.microsoft.com/Microsoft-Band/`.

[Mietus et al., 2002] Mietus, J. E., Peng, C.-K., Henry, I., Goldsmith, R. L., and Goldberger, A. L. (2002). The pNNx files: re-examining a widely used heart rate variability measure. *Heart*, 88(4):378–380.

[Moha et al., 2010] Moha, N., Guéhéneuc, Y., Duchien, L., and Le Meur, A. (2010). Decor: A method for the specification and detection of code and design smells. *Transactions on Software Engineering*, 36(1).

[Moser et al., 2008] Moser, R., Pedrycz, W., and Succi, G. (2008). Analysis of the reliability of a subset of change metrics for defect prediction. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, pages 309–311.

[Muldner et al., 2010] Muldner, K., Burleson, W., and VanLehn, K. (2010). "Yes!": using tutor and sensor data to predict moments of delight during instructional activities. In *Proceedings of the 18th International Conference on User Modeling, Adaptation, and Personalization*, pages 159–170.

[Muldner et al., 2009] Muldner, K., Christopherson, R., Atkinson, R., and Burleson, W. (2009). Investigating the utility of eye-tracking information on affect and reasoning for user modeling. In *Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization*, pages 138–149.

[Müller and Fritz, 2015a] Müller, S. C. and Fritz, T. (2015a). Replication package. Available online. `http://www.ifi.uzh.ch/seal/people/mueller/PredictCodeQualityWithBiometrics`.

[Müller and Fritz, 2015b] Müller, S. C. and Fritz, T. (2015b). Stuck and frustrated or in flow and happy: Sensing developers' emotions and progress. In *Proceedings of the International Conference on Software Engineering*, pages 688 – 699.

[Müller and Fritz, 2016] Müller, S. C. and Fritz, T. (2016). Using (bio)metrics to predict code quality online. In *Proceedings of International Conference on Software Engineering*, page 12 pages to appear.

[Munro, 2005] Munro, M. (2005). Product metrics for automatic identification of "bad smell" design problems in java source-code. In *Proceedings of the International Symposium on Software Metrics*, page 15.

[Murugappan et al., 2009] Murugappan, M., Nagarajan, R., and Yaacob, S. (2009). Modified energy based time-frequency features for classifying human emotions using eeg. In *Proceedings of the International Conference on Man-Machine Systems*, page N/A.

[Murugappan et al., 2008] Murugappan, M., Rizon, M., Nagarajan, R., and Yaacob, S. (2008). Eeg feature extraction for classifying emotions using fcm and fkm. In *Proceedings of the 7th WSEAS International Conference on Applied Computer and Applied Computational Science*, pages 299–304.

[Nagappan and Ball, 2005] Nagappan, N. and Ball, T. (2005). Use of relative code churn measures to predict system defect density. In *Proceedings of International Conference on Software Engineering*, pages 284–292.

[Nagappan et al., 2006] Nagappan, N., Ball, T., and Zeller, A. (2006). Mining metrics to predict component failures. In *Proceedings of International Conference on Software Engineering*, pages 452–461.

[Nagappan et al., 2008] Nagappan, N., Murphy, B., and Basili, V. (2008). The influence of organizational structure on software quality: An empirical case study. In *Proceedings of the International Conference on Software Engineering*, pages 521–530.

[Nakagawa et al., 2014] Nakagawa, T., Kamei, Y., Uwano, H., Monden, A., Matsumoto, K., and German, D. M. (2014). Quantifying programmers' mental workload during program comprehension based on cerebral blood flow measurement: A controlled experiment. In *Companion Proceedings of International Conference on Software Engineering*, pages 448–451.

[Nakasone et al., 2005] Nakasone, A., Prendinger, H., and Ishizuka, M. (2005). Emotion recognition from electromyography and skin conductance. In *Proceedings of the 5th International Workshop on Biosignal Interpretation*, pages 1210 – 1215.

[NASA - Ames Research Center, Aerospace Human Factors Research Division, 1986] NASA - Ames Research Center, Aerospace Human Factors Research Division (1986). NASA TLX Load Index (TLX): Paper-and-pencil version.

[Neurosky, 2009] Neurosky (2009). Neurosky's eSense™ meters and detection of mental state.

[Neurosky, 2015] Neurosky (2015). Neurosky Mindband. `http://neurosky.com/biosensors/eeg-sensor/`.

[Nevalainen and Sajaniemi, 2005] Nevalainen, S. and Sajaniemi, J. (2005). Short-term effects of graphical versus textual visualisation of variables on program perception. In *Proceedings of the 17th Annual Psychology of Prgramming Interest Group Workshop*, pages 77–91.

[Nike+ Fuelband, 2015] Nike+ Fuelband (2015). `http://www.nike.com/us/en_us/c/nikeplus-fuel`.

[Norman, 2010] Norman, G. (2010). Likert scales, levels of measurement and the "laws" of statistics. *Advances in Health Science Education: Theory and Practice*, 15(5):625–632.

[Nourbakhsh et al., 2012] Nourbakhsh, N., Wang, Y., Chen, F., and Calvo, R. A. (2012). Using galvanic skin response for cognitive load measurement in arithmetic and reading tasks. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*, pages 420–423.

[Novak et al., 2010] Novak, D., Ziherl, J., Olenšek, A., Milavec, M., Podobnik, J., Mihelj, M., and Munih, M. (2010). Psychophysiological response to robotic rehabilitation tasks in stroke. *Transactions on Neural Systems and Rehabilitation Engineering*, 18(4).

[Palomba et al., 2013] Palomba, F., Bavota, G., Di Penta, M., Oliveto, R., De Lucia, A., and Poshyvanyk, D. (2013). Detecting bad smells in source code using change history information. In *International Conference on Automated Software Engineering*, pages 268 – 278.

[Parnin, 2011] Parnin, C. (2011). Subvocalization - toward hearing the inner thoughts of developers. In *Proceedings of the International Conference on Program Comprehension*, pages 197–200.

[Peper et al., 2007] Peper, E., Harvey, R., Lin, I.-M., Tylova, H., and Moss, D. (2007). Is there more to blood volume pulse than heart rate variability respiratory sinus arrhythmia, and cardiorespiratory synchrony? *Biofeedback*, 35(2):54–61.

[Picard et al., 2001] Picard, R. W., Vyzas, E., and Healey, J. (2001). Toward machine emotional intelligence: Analysis of affective physiological state. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1175–1191.

[Plass et al., 2010] Plass, J. L., Moreno, R., and Brünken, R., editors (2010). *Cognitive Load Theory*. Cambridge University Press.

[Plutchik and Conte, 1997] Plutchik, R. and Conte, H. R. (1997). *Circumplex Models of Personality and Emotions*. American Psychological Association.

[PMD, 2015] PMD (2015). `https://pmd.github.io/`.

[Poh et al., 2010] Poh, M.-Z., Swenson, N., and Picard, R. (2010). A wearable sensor for unobtrusive, long-term assessment of electrodermal activity. *IEEE Transactions on Biomedical Engineering*, 57(5):1243–1252.

[Poole and Ball, 2006] Poole, A. and Ball, L. J. (2006). *Encyclpedia of Human Computer Interaction*, chapter Eye Tracking in HCI and Usability Research. Idea Group Reference.

[Porras and Guéhéneuc, 2010] Porras, G. C. and Guéhéneuc, Y.-G. (2010). An empirical study on the efficiency of different design pattern representations in UML class diagrams. *Empirical Software Engineering*, 15(5):493–522.

[Qi, 2012] Qi, Y. (2012). Random forest for bioinformatics. In Zhang, C. and Ma, Y., editors, *Ensemble Machine Learning*, pages 307–323. Springer US.

[Quinlan, 1993] Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA.

[Radevski et al., 2015] Radevski, S., Hata, H., and Matsumoto, K. (2015). Real-time monitoring of neural state in assessing and improving software developers' productivity. *Proceedings of the International Workshop on Collaborative and Human Aspects of Software Engineering*, pages 93–96.

[Rani et al., 2004] Rani, P., Sarkar, N., Smith, C. A., and Kirby, L. D. (2004). Anxiety detecting robotic system - towards implicit human-robot collaboration. *Robotica*, 22(1):85–95.

[Rayner, 1998] Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychological bulletin*, 124(3):372.

[Reuderink et al., 2013] Reuderink, B., Mühl, C., and Poel, M. (2013). Valence, arousal and dominance in the eeg during game play. *International Journal of Autonomous and Adaptive Communications Systems*, 6(1):45–62.

[Reuderink et al., 2009] Reuderink, B., Nijholt, A., and Poel, M. (2009). Affective pacman: A frustrating game for brain-computer interface experiments. In

Nijholt, A., Reidsma, D., and Hondorp, H., editors, *Intelligent Technologies for Interactive Entertainment*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer.

[Richter et al., 1998] Richter, P., Wagner, T., Heger, R., and Weise, G. (1998). Psychophysiological analysis of mental load during driving on rural roads - a quasi-experimental field study. *Ergonomics*, 41(5).

[Rieger et al., 2014] Rieger, A., Stoll, R., Kreuzfeld, S., Behrens, K., and Weippert, M. (2014). Heart rate and heart rate variability as indirect markers of surgeons' intraoperative stress. *International Archives of Occupational and Environmental Health*, 87(2):165–174.

[Rigby et al., 2008] Rigby, P. C., German, D. M., and Storey, M.-A. (2008). Open source software peer review practices: A case study of the apache server. In *Proceedings of International Conference on Software Engineering*, pages 541–550.

[Riseberg et al., 1998] Riseberg, J., Klein, J., Fernandez, R., and Picard, R. W. (1998). Frustrating the user on purpose: Using biosignals in a pilot study to detect the user's emotional state. In *Conference Summary on Human Factors in Computing Systems*, pages 227–228.

[Rodeghero et al., 2014] Rodeghero, P., McMillan, C., McBurney, P. W., Bosch, N., and D'Mello, S. (2014). Improving automated source code summarization via an eye-tracking study of programmers. In *Proceedings of International Conference on Software Engineering*, pages 390–401.

[Rowe et al., 1998] Rowe, D. W., Sibert, J., and Irwin, D. (1998). Heart rate variability: Indicator of user state as an aid to human-computer interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 480–487.

[Russell, 1980] Russell, J. (1980). A circumplex model of affect. *Journal of Personality and Social Psychology*, 39(6):1161–1178.

[Russell, 2003] Russell, J. A. (2003). Core affect and the psychological construction of emotion. *Psychological Review*, 110(1):145–172.

[Ryu and Myung, 2005] Ryu, K. and Myung, R. (2005). Evaluation of mental workload with a combined measure based on physiological indices during a dual task of tracking and mental arithmetic. *International Journal of Industrial Ergonomics*, 35(11):991 – 1009.

[Sammler et al., 2007] Sammler, D., Grigutsch, M., Fritz, T., and Koelsch, S. (2007). Music and emotion: electrophysiological correlates of the processing of pleasant and unpleasant music. *Psychophysiology*, 44(2):293–304.

[Scheirer et al., 2002] Scheirer, J., Fernandez, R., Klein, J., and Picard, R. W. (2002). Frustrating the user on purpose: a step toward building an affective computer. *Interacting with Computers*, 14(2):93 – 118.

[Schmidt and Walach, 2000] Schmidt, S. and Walach, H. (2000). Electrodermal activity (EDA) - state-of-the-art measurements and techniques for parapsychological purposes. *Journal of Parapsychology*, 64(2):139.

[Setz et al., 2010] Setz, C., Arnrich, B., Schumm, J., La Marca, R., Tröster, G., and Ehlert, U. (2010). Discriminating stress from cognitive load using a wearable EDA device. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):410–417.

[Sharafi et al., 2013] Sharafi, Z., Marchetto, A., Susi, A., and Antoniol, G. (2013). An empirical study on the efficiency of graphical vs. textual representations in requirements comprehension. In *Proceedings of the International Conference on Program Comprehension*, pages 33–42.

[Sharafi et al., 2012] Sharafi, Z., Soh, Z., Guéhéneuc, Y.-G., and Antoniol, G. (2012). Women and men - different but equal: On the impact of identifier style on source code reading. In *Proceedings of the 20th International Conference on Program Comprehension*, pages 27–36.

[Sharif et al., 2012] Sharif, B., Falcone, M., and Maletic, J. I. (2012). An eye-tracking study on the role of scan time in finding source code defects. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 381–384.

[Sharif and Maletic, 2010a] Sharif, B. and Maletic, J. I. (2010a). An eye tracking study on camelcase and under_score identifier styles. In *Proceedings of the 18th International Conference on Program Comprehension*, pages 196–205.

[Sharif and Maletic, 2010b] Sharif, B. and Maletic, J. I. (2010b). An eye tracking study on the effects of layout in understanding the role of design patterns. In *Proceedings of the International Conference on Software Maintenance*, pages 1–10.

[Shaw, 2004] Shaw, T. (2004). The emotions of systems developers: An empirical study of affective events theory. In *Proceedings of the Conference on Computer Personnel Research: Careers, Culture, and Ethics in a Networked Environment*, pages 124–126.

[Shi et al., 2007] Shi, Y., Ruiz, N., Taib, R., Choi, E., and Chen, F. (2007). Galvanic skin response (GSR) as an index of cognitive load. In *CHI '07 Extended Abstracts on Human Factors in Computing Systems*, pages 2651–2656.

[Siegmund et al., 2014] Siegmund, J., Kästner, C., Apel, S., Parnin, C., Bethmann, A., Leich, T., Saake, G., and Brechmann, A. (2014). Understanding understanding source code with functional magnetic resonance imaging. In *Proceedings of the 36th International Conference on Software Engineering*, pages 378–389.

[Sillito et al., 2005] Sillito, J., De Voider, K., Fisher, B., and Murphy, G. (2005). Managing software change tasks: an exploratory study. In *International Symposium on Empirical Software Engineering*, pages 23–32.

[Simola et al., 2008] Simola, J., Salojärvi, J., and Kojo, I. (2008). Using hidden markov model to uncover processing states from eye movements in information search tasks. *Cognitive Systems Research*, 9(4):237–251.

[Smith and Gevins, 2005] Smith, M. E. and Gevins, A. (2005). Neurophysiologic monitoring of mental workload and fatigue during operation of a flight simulator. In *Defense and Security*, pages 116–126.

[Spector, 1990] Spector, R. H. (1990). *Clinical Methods: The history, physical and laboratory examinations*, chapter The Pupils. Butterworths.

[Sroufe and Waters, 1977] Sroufe, L. A. and Waters, E. (1977). Heart rate as a convergent measure in clinical and developmental research. *Merrill-Palmer Quarterly of Behavior and Development*, 23(1):3–27.

[StackExchange, 2014] StackExchange (2014). `http://api.stackexchange.com/`.

[Steptoe et al., 2005] Steptoe, A., Wardle, J., and Marmot, M. (2005). Positive affect and health-related neuroendocrine, cardiovascular, and inflammatory processes. *Proceedings of the National Academy of Sciences*, 102(18):6508–6512.

[Sterman et al., 1993] Sterman, M., Mann, C., and Kaiser, D. (1993). Quantitative EEG patterns of differential in-flight workload. In *Proceedings of the 6th Annual Workshop on Space Operations Applications and Research*, pages 466–473.

[Storey et al., 2014] Storey, M.-A., Singer, L., Cleary, B., Figueira Filho, F., and Zagalsky, A. (2014). The (r)evolution of social media in software engineering. In *Proceedings of the on Future of Software Engineering*, pages 100–116.

[Sweller, 1988] Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2):257–285.

[Sweller et al., 2011] Sweller, J., Ayres, P., and Kalyuga, S. (2011). *Cognitive Load Theory*. Springer.

[Tani and Yamada, 2013] Tani, T. and Yamada, S. (2013). Estimating user interruptibility by measuring table-top pressure. In *Extended Abstracts on Human Factors in Computing Systems*, pages 1707–1712.

[Techcrunch, 2015] Techcrunch (2015). Meet phabricator, the witty code review tool built inside facebook. `http://techcrunch.com/2011/08/07/oh-what-noble-scribe-hath-penned-these-words/`.

[Telford and Thompson, 1933] Telford, C. and Thompson, N. (1933). Some factors influencing voluntary and reflex eyelid responses. *Journal of Experimental Psychology*, 16(4):524.

[Tiarks and Roehm, 2012] Tiarks, R. and Roehm, T. (2012). Challenges in program comprehension. *Softwaretechnik-Trends*, 32(2).

[Tobii, 2015] Tobii (2015). Tobii Pro TX300. `http://www.tobiipro.com/product-listing/tobii-pro-tx300/`.

[Torii et al., 1999] Torii, K., Matsumoto, K.-i., Nakakoji, K., Takada, Y., Takada, S., and Shima, K. (1999). Ginger2: An environment for computer-aided empirical software engineering. *IEEE Transactions on Software Engineering*, 25(4):474–492.

[University of Phoenix, 2015] University of Phoenix (2015). Building software development talent steps for a sustainable future. Survey Summary.

[Uwano et al., 2006] Uwano, H., Nakamura, M., Monden, A., and Matsumoto, K.-I. (2006). Analyzing individual performance of source code review using reviewers' eye movement. In *Proceedings of the 2006 Symposium on Eye Tracking Research & Applications*, pages 133–140.

[van Emden and Moonen, 2002] van Emden, E. and Moonen, L. (2002). Java quality assurance by detecting code smells. In *Proceedings of the Working Conference on Reverse Engineering*, pages 97 – 106.

[Veltman and Gaillard, 1998] Veltman, J. and Gaillard, A. W. (1998). Physiological workload reactions to increasing levels of task difficulty. *Ergonomics*, 41(5):656–669.

[Wagner et al., 2005] Wagner, J., Kim, J., and Andre, E. (2005). From physiological signals to emotions: Implementing and comparing selected methods for feature extraction and classification. In *Proceedings of the International Conference on Multimedia & Expo*, pages 940–943.

[Walter and Porges, 1976] Walter, G. F. and Porges, S. W. (1976). Heart rate and respiratory responses as a function of task difficulty: The use of discriminant analysis in the selection of psychologically sensitive physiological responses. *Psychophysiology*, 13(6).

[Walters et al., 2013] Walters, B., Falcone, M., Shibble, A., and Sharif, B. (2013). Towards an eye-tracking enabled ide for software traceability tasks. In *Workshop on Traceability in Emerging Forms of Software Engineering*, pages 51–54.

[Walters et al., 2014] Walters, B., Shaffer, T., Sharif, B., and Kagdi, H. (2014). Capturing software traceability links from developers' eye gazes. In *Proceedings of the International Conference on Program Comprehension*, pages 201–204.

[Weast and Neiman, 2010] Weast, R. A. and Neiman, N. G. (2010). The effect of cognitive load and meaning on selective attention. In *Annual Meeting of the Cognitive Science Society*, pages 1477–1482.

[Weyuker et al., 2008] Weyuker, E. J., Ostrand, T. J., and Bell, R. M. (2008). Do too many cooks spoil the broth? using the number of developers to enhance defect prediction models. *Empirical Software Engineering*, 13(5):539–559.

[Wilson, 1992] Wilson, G. F. (1992). Applied use of cardiac and respiration measures: Practical considerations and precautions. *Biological Psychology*, 34(2–3):163 – 178.

[Wilson, 2002] Wilson, G. F. (2002). An Analysis of Mental Workload in Pilots During Flight Using Multiple Psychophysiological Measures. *International Journal of Aviation Psychology*, 12(1):3–18.

[Woodworth, 1948] Woodworth (1948). *Experimental Psychology*. Holt.

[Wrobel, 2013] Wrobel, M. (2013). Emotions in the software development process. In *Proceedings of the International Conference on Human System Interaction*, pages 518–523.

[Yoon et al., 2013] Yoon, H., Park, S.-W., Lee, Y.-K., and Jang, J.-H. (2013). Emotion recognition of serious game players using a simple brain computer interface. In *International Conference on ICT Convergence*, pages 783–786.

[Zhang et al., 2007] Zhang, H., Zhang, X., and Gu, M. (2007). Predicting defective software components from code complexity measures. In *Proceedings of the Pacific Rim International Symposium on Dependable Computing*, pages 93 – 96.

[Zimmermann et al., 2007] Zimmermann, T., Premraj, R., and Zeller, A. (2007). Predicting defects for eclipse. In *Proceedings of the Workshop on Predictor Models in Software Engineering*, page 9.

[Züger and Fritz, 2015] Züger, M. and Fritz, T. (2015). Interruptibility of software developers and its prediction using psycho-physiological sensors. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 2981–2990.

Curriculum Vitae

Personal Information

Name	Sebastian Christoph Müller
Nationality	Liechtenstein
Date of Birth	November 6, 1985
Place of Birth	Vaduz, Liechtenstein

Education

2011 – 2016	PhD in Informatics Department of Informatics University of Zurich, Switzerland
2009 – 2011	Master of Science in Computer Science Department of Informatics University of Zurich, Switzerland
2005 – 2009	Bachelor of Science in Informatics Department of Informatics University of Zurich, Switzerland